The VGAM Package

October 1, 2007

Version 0.7-4

Date 2007-10-1

Title Vector Generalized Linear and Additive Models

Author Thomas W. Yee <t.yee@auckland.ac.nz>

Maintainer Thomas Yee <t.yee@auckland.ac.nz>

Depends R (>= 2.4.0), splines, methods, stats, stats4

Description Vector generalized linear and additive models, and associated models (Reduced-Rank VGLMs, Quadratic RR-VGLMs, Reduced-Rank VGAMs). This package fits many models and distribution by maximum likelihood estimation (MLE) or penalized MLE. Also fits constrained ordination models in ecology.

License GPL version 2

URL http://www.stat.auckland.ac.nz/~yee/VGAM

LazyLoad yes

LazyData yes

R topics documented:

AA.Aa.aa *The AA-Aa-aa Blood Group System*

Description

Estimates the parameter of the AA-Aa-aa blood group system.

Usage

```
AA.Aa.aa(link = "logit", earg=list(), init.pA = NULL)
```

Arguments

link	Link function applied to pA. See Links for more choices.
earg	List. Extra argument for the link. See earg in Links for general information.
init.pA	Optional initial value for pA.

Details

This one parameter model involves a probability called pA. The probability of getting a count in the first column of the input (an AA) is pA*pA.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

The input can be a 3-column matrix of counts, where the columns are AA, Ab and aa (in order). Alternatively, the input can be a 3-column matrix of proportions (so each row adds to 1) and the weights argument is used to specify the total number of counts for each row.

Author(s)

T. W. Yee

References

Weir, B. S. (1996) *Genetic Data Analysis II: Methods for Discrete Population Genetic Data*, Sunderland, MA: Sinauer Associates, Inc.

See Also

AB.Ab.aB.ab, AB.Ab.aB.ab2, ABO, G1G2G3, MNSs.

Examples

```
y = cbind(53, 95, 38)
fit = vglm(y ~ 1, AA.Aa.aa(link="logit"), trace=TRUE)
fit = vglm(y ~ 1, AA.Aa.aa(link="probit"), trace=TRUE)
rbind(y, sum(y)*fitted(fit))
Coef(fit) # Estimated pA
summary(fit)
```

`AB.Ab.aB.ab` *The AB-Ab-aB-ab Blood Group System*

Description

Estimates the parameter of the AB-Ab-aB-ab blood group system.

Usage

```
AB.Ab.aB.ab(link = "logit", earg=list(), init.p = NULL)
```

Arguments

`link`	Link function applied to `p`. See `Links` for more choices.
`earg`	List. Extra argument for the link. See `earg` in `Links` for general information.
`init.p`	Optional initial value for `p`.

Details

This one parameter model involves a probability called `p`.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Note

The input can be a 4-column matrix of counts, where the columns are AB, Ab, aB and ab (in order). Alternatively, the input can be a 4-column matrix of proportions (so each row adds to 1) and the `weights` argument is used to specify the total number of counts for each row.

Author(s)

T. W. Yee

References

Lange, K. (2002) *Mathematical and Statistical Methods for Genetic Analysis*, 2nd ed. New York: Springer-Verlag.

See Also

AA.Aa.aa, AB.Ab.aB.ab2, ABO, G1G2G3, MNSs.

Examples

```
y = cbind(1997, 906, 904, 32) # Data from Fisher (1925)
fit = vglm(y ~ 1, AB.Ab.aB.ab(link="identity", init.p=0.9), trace=TRUE)
fit = vglm(y ~ 1, AB.Ab.aB.ab, trace=TRUE)
rbind(y, sum(y)*fitted(fit))
Coef(fit) # Estimated p
p = sqrt(4*(fitted(fit)[,4]))
p*p
summary(fit)
```

AB.Ab.aB.ab2 *The AB-Ab-aB-ab2 Blood Group System*

Description

Estimates the parameter of the the AB-Ab-aB-ab2 blood group system.

Usage

```
AB.Ab.aB.ab2(link = "logit", earg=list(), init.p = NULL)
```

Arguments

link	Link function applied to p. See Links for more choices.
earg	List. Extra argument for the link. See earg in Links for general information.
init.p	Optional initial value for p.

Details

This one parameter model involves a probability called p.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Warning

There may be a bug in the deriv and weight slot of the family function.

Note

The input can be a 4-column matrix of counts. Alternatively, the input can be a 4-column matrix of proportions (so each row adds to 1) and the weights argument is used to specify the total number of counts for each row.

Author(s)

T. W. Yee

References

Elandt-Johnson, R. C. (1971) *Probability Models and Statistical Methods in Genetics*, New York: Wiley.

See Also

AA.Aa.aa, AB.Ab.aB.ab, ABO, G1G2G3, MNSs.

Examples

```
# See Elandt-Johnson, pp.430,427
# Estimated variance is approx 0.0021
y = cbind(68, 11, 13, 21)
fit = vglm(y ~ 1, AB.Ab.aB.ab2(link=logit), trace=TRUE, crit="coef")
fit = vglm(y ~ 1, AB.Ab.aB.ab2(link=cloglog), trace=TRUE, crit="coef")
Coef(fit)   # Estimated p
rbind(y, sum(y)*fitted(fit))
diag(vcov(fit))^0.5
```

ABO *The ABO Blood Group System*

Description

Estimates the two independent parameters of the the ABO blood group system.

Usage

```
ABO(link = "logit", earg=list(), ir = NULL, ip = NULL)
```

Arguments

link	Link function applied to p and q. See Links for more choices.
earg	List. Extra argument applied to each of the links. See earg in Links for general information.
ir, ip	Optional initial value for r and p. A NULL value means values are computed internally.

Details

The parameters p and q are probabilities, so that r=1-p-q is the third probability. The probabilities p and r correspond to A and O respectively, so that q is the probability for B. It is easier to make use of initial values for r than for q.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

The input can be a 4-column matrix of counts, where the columns are A, B, AB, O (in order). Alternatively, the input can be a 4-column matrix of proportions (so each row adds to 1) and the weights argument is used to specify the total number of counts for each row.

Author(s)

T. W. Yee

References

Lange, K. (2002) *Mathematical and Statistical Methods for Genetic Analysis*, 2nd ed. New York: Springer-Verlag.

See Also

AA.Aa.aa, AB.Ab.aB.ab, AB.Ab.aB.ab2, G1G2G3, MNSs.

Examples

```
y = cbind(A=725, B=258, AB=72, O=1073) # Order matters, not the name
fit = vglm(y ~ 1, ABO(link=logit), trace=TRUE, cri="coef")
fit = vglm(y ~ 1, ABO(link=identity), trace=TRUE, cri="coef")
coef(fit, matrix=TRUE)
Coef(fit) # Estimated p and q
rbind(y, sum(y)*fitted(fit))
sqrt(diag(vcov(fit)))
```

Brat *Inputting Data to fit a Bradley Terry Model*

Description

Takes in a square matrix of counts and outputs them in a form that is accessible to the brat and bratt family functions.

Usage

```
Brat(mat, ties=0*mat, string=c(" > "," == "))
```

Arguments

`mat`	Matrix of counts, which is considered M by M in dimension when there are ties, and $M+1$ by $M+1$ when there are no ties. The rows are winners and the columns are losers, e.g., the 2-1 element is now many times Competitor 2 has beaten Competitor 1. The matrices are best labelled with the competitors' names.
`ties`	Matrix of counts. This should be the same dimension as `mat`. By default, there are no ties. The matrix must be symmetric, and the diagonal should contain `NA`s.
`string`	Character. The matrices are labelled with the first value of the descriptor, e.g., `"NZ > Oz"` 'means' NZ beats Australia in rugby. Suggested alternatives include `" beats "` or `" wins against "`. The second value is used to handle ties.

Details

In the **VGAM** package it is necessary for each matrix to be represented as a single row of data by `brat` and `bratt`. Hence the non-diagonal elements of the $M+1$ by $M+1$ matrix are concatenated into $M(M+1)$ values (no ties), while if there are ties, the non-diagonal elements of the M by M matrix are concatenated into $M(M-1)$ values.

Value

A matrix with 1 row and either $M(M+1)$ or $M(M-1)$ columns.

Note

This is a data preprocessing function for `brat` and `bratt`.

Yet to do: merge `InverseBrat` into `brat`.

Author(s)

T. W. Yee

References

Agresti, A. (2002) *Categorical Data Analysis*, 2nd ed. New York: Wiley.

See Also

`brat`, `bratt`, `InverseBrat`.

Examples

```
journal = c("Biometrika", "Comm Statist", "JASA", "JRSS-B")
m = matrix(c( NA, 33, 320, 284,   730, NA, 813, 276,
             498, 68,  NA, 325,   221, 17, 142, NA), 4,4)
dimnames(m) = list(winner = journal, loser = journal)
Brat(m)
vglm(Brat(m) ~ 1, brat, trace=TRUE)
```

Coef *Computes Model Coefficients and Quantities*

Description

Coef is a generic function which computes model coefficients from objects returned by modelling functions. It is an auxiliary function to coef that enables extra capabilities for some specific models.

Usage

```
Coef(object, ...)
```

Arguments

object	An object for which the computation of other types of model coefficients or quantities is meaningful.
...	Other arguments fed into the specific methods function of the model.

Details

This function can often be useful for vglm objects with just an intercept term in the RHS of the formula, e.g., y ~ 1. Then often this function will apply the inverse link functions to the parameters. See the example below.

For reduced-rank VGLMs, this function can return the **A**, **C** matrices, etc.

For quadratic and additive ordination models, this function can return ecological meaningful quantities such as tolerances, optima, maxima.

Value

The value returned depends specifically on the methods function invoked.

Warning

This function may not work for *all* **VGAM** family functions. You should check your results on some artificial data before applying it to models fitted to real data.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

See Also

coef, Coef.vlm, Coef.rrvglm, Coef.qrrvglm.

Examples

```
set.seed(123)
nn = 1000
y = rbeta(nn, shape1=1, shape2=3) # Original scale
# parameters are estimated on a log scale:
fit = vglm(y ~ 1, betaff, tr=TRUE, crit="c") # intercept-only model
coef(fit, matrix=TRUE) #  log scale
Coef(fit) # On the original scale
```

Coef.qrrvglm-class *Class "Coef.qrrvglm"*

Description

The most pertinent matrices and other quantities pertaining to a QRR-VGLM (CQO model).

Objects from the Class

Objects can be created by calls of the form `Coef(object, ...)` where `object` is an object of class `"qrrvglm"` (created by cqo).

In this document, R is the *rank*, M is the number of linear predictors and n is the number of observations.

Slots

A: Of class `"matrix"`, **A**, which are the linear 'coefficients' of the matrix of latent variables. It is M by R.

B1: Of class `"matrix"`, **B1**. These correspond to terms of the argument `Norrr`.

C: Of class `"matrix"`, **C**, the canonical coefficients. It has R columns.

Constrained: Logical. Whether the model is a constrained ordination model.

D: Of class `"array"`, `D[,,j]` is an order-`Rank` matrix, for `j` = 1,...,M. Ideally, these are negative-definite in order to make the response curves/surfaces bell-shaped.

Rank: The rank (dimension, number of latent variables) of the RR-VGLM. Called R.

lv: n by R matrix of latent variable values.

lvOrder: Of class `"matrix"`, the permutation returned when the function order is applied to each column of `lv`. This enables each column of `lv` to be easily sorted.

Maximum: Of class `"numeric"`, the M maximum fitted values. That is, the fitted values at the optima for `Norrr = ~ 1` models. If `Norrr` is not `~ 1` then these will be `NA`s.

NOS: Number of species.

Optimum: Of class "matrix", the values of the latent variables where the optima are. If the curves are not bell-shaped, then the value will be NA or NaN.

OptimumOrder: Of class "matrix", the permutation returned when the function order is applied to each column of Optimum. This enables each row of Optimum to be easily sorted.

bellshaped: Vector of logicals: is each response curve/surface bell-shaped?

dispersion: Dispersion parameter(s).

Dzero: Vector of logicals, is each of the response curves linear in the latent variable(s)? It will be if and only if D[,,j] equals **O**, for j = 1,...,M .

Tolerance: Object of class "array", Tolerance[,,j] is an order-Rank matrix, for j = 1,...,M, being the matrix of tolerances (squared if on the diagonal). These are denoted by **T** in Yee (2004). Ideally, these are positive-definite in order to make the response curves/surfaces bell-shaped. The tolerance matrices satisfy $T_s = -\frac{1}{2} D_s^{-1}$.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2004) A new technique for maximum-likelihood canonical Gaussian ordination. *Ecological Monographs*, **74**, 685–701.

See Also

Coef.qrrvglm, cqo, print.Coef.qrrvglm.

Examples

```
x2 = rnorm(n <- 100)
x3 = rnorm(n)
x4 = rnorm(n)
lv1 = 0 + x3 - 2*x4
lambda1 = exp(3 - 0.5 * (lv1-0)^2)
lambda2 = exp(2 - 0.5 * (lv1-1)^2)
lambda3 = exp(2 - 0.5 * ((lv1+4)/2)^2)
y1 = rpois(n, lambda1)
y2 = rpois(n, lambda2)
y3 = rpois(n, lambda3)
yy = cbind(y1,y2,y3)
p1 = cqo(yy ~ x2 + x3 + x4, fam=poissonff, trace=FALSE)
## Not run:
lvplot(p1, y=TRUE, lcol=1:3, pch=1:3, pcol=1:3)
## End(Not run)
print(Coef(p1), digits=3)
```

`Coef.qrrvglm` *Returns Important Matrices etc. of a QO Object*

Description

This methods function returns important matrices etc. of a QO object.

Usage

```
Coef.qrrvglm(object, varlvI = FALSE, reference = NULL, ...)
```

Arguments

`object`	A CQO or UQO object. The former has class `"qrrvglm"`.
`varlvI`	Logical indicating whether to scale the site scores (latent variables) to have variance-covariance matrix equal to the rank-R identity matrix. All models have uncorrelated site scores (latent variables), and this option stretches or shrinks the ordination axes if `TRUE`. See below for further details.
`reference`	Integer or character. Specifies the *reference species*. By default, the reference species is found by searching sequentially starting from the first species until a positive-definite tolerance matrix is found. Then this tolerance matrix is transformed to the identity matrix. Then the sites scores (latent variables) are made uncorrelated. See below for further details.
`...`	Currently unused.

Details

If `ITolerances=TRUE` or `EqualTolerances=TRUE` (and its estimated tolerance matrix is positive-definite) then all species' tolerances are unity by transformation or by definition, and the spread of the site scores can be compared to them. Vice versa, if one wishes to compare the tolerances with the sites score variability then setting `varlvI=TRUE` is more appropriate.

For rank-2 QRR-VGLMs, one of the species can be chosen so that the angle of its major axis and minor axis is zero, i.e., parallel to the ordination axes. This means the effect on the latent vars is independent on that species, and that its tolerance matrix is diagonal. The argument `reference` allows one to choose which is the reference species, which must have a positive-definite tolerance matrix, i.e., is bell-shaped. If `reference` is not specified, then the code will try to choose some reference species starting from the first species. Although the `reference` argument could possibly be offered as an option when fitting the model, it is currently available after fitting the model, e.g., in the functions `Coef.qrrvglm` and `lvplot.qrrvglm`.

Value

The **A**, **B1**, **C**, **T**, **D** matrices/arrays are returned, along with other slots. For UQO, **C** is undefined. The returned object has class `"Coef.qrrvglm"` (see `Coef.qrrvglm-class`).

Note

Consider an equal-tolerances Poisson/binomial CQO model with `Norrr = ~ 1`. For $R = 1$ it has about $2S + p_2$ parameters. For $R = 2$ it has about $3S + 2p_2$ parameters. Here, S is the number of species, and $p_2 = p - 1$ is the number of environmental variables making up the latent variable. For an unequal-tolerances Poisson/binomial CQO model with `Norrr = ~ 1`, it has about $3S - 1 + p_2$ parameters for $R = 1$, and about $6S - 3 + 2p_2$ parameters for $R = 2$. Since the total number of data points is nS, where n is the number of sites, it pays to divide the number of data points by the number of parameters to get some idea about how much information the parameters contain.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2004) A new technique for maximum-likelihood canonical Gaussian ordination. *Ecological Monographs*, **74**, 685–701.

Yee, T. W. (2006) Constrained additive ordination. *Ecology*, **87**, 203–213.

See Also

`cqo`, `Coef.qrrvglm-class`, `print.Coef.qrrvglm`, `lvplot.qrrvglm`.

Examples

```
set.seed(123)
x2 = rnorm(n <- 100)
x3 = rnorm(n)
x4 = rnorm(n)
lv1 = 0 + x3 - 2*x4
lambda1 = exp(3 - 0.5 * (lv1-0)^2)
lambda2 = exp(2 - 0.5 * (lv1-1)^2)
lambda3 = exp(2 - 0.5 * ((lv1+4)/2)^2) # Unequal tolerances
y1 = rpois(n, lambda1)
y2 = rpois(n, lambda2)
y3 = rpois(n, lambda3)
set.seed(111)
p1 = cqo(cbind(y1,y2,y3) ~ x2 + x3 + x4, poissonff, trace=FALSE)
## Not run:
lvplot(p1, y=TRUE, lcol=1:3, pch=1:3, pcol=1:3)
## End(Not run)
Coef(p1)
print(Coef(p1), digits=3)
```

`Coef.rrvglm-class` *Class "Coef.rrvglm"*

Description

The most pertinent matrices and other quantities pertaining to a RR-VGLM.

Objects from the Class

Objects can be created by calls of the form `Coef(object, ...)` where `object` is an object of class `rrvglm` (see `rrvglm-class`).

In this document, M is the number of linear predictors and n is the number of observations.

Slots

`A`: Of class `"matrix"`, **A**.

`B1`: Of class `"matrix"`, **B1**.

`C`: Of class `"matrix"`, **C**.

`Rank`: The rank of the RR-VGLM.

`colx1.index`: Index of the columns of the `"vlm"`-type model matrix corresponding to the variables in **x1**. These correspond to **B1**.

`colx2.index`: Index of the columns of the `"vlm"`-type model matrix corresponding to the variables in **x2**. These correspond to the reduced-rank regression.

`Atilde`: Object of class `"matrix"`, the **A** matrix with the corner rows removed. Thus each of the elements have been estimated. This matrix is returned only if corner constraints were used.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

See Also

`Coef.rrvglm`, `rrvglm`, `rrvglm-class`, `print.Coef.rrvglm`.

Examples

```
# Rank-1 stereotype model of Anderson (1984)
data(pneumo)
n = nrow(pneumo)
pneumo = transform(pneumo, let=log(exposure.time), x1=runif(n), x2=runif(n))
fit = rrvglm(cbind(normal,mild,severe) ~ let + x1 + x2, multinomial, pneumo)
coef(fit, mat=TRUE)
Coef(fit)
print(Coef(fit), digits=3)
```

Coef.rrvglm *Returns Important Matrices etc. of a RR-VGLM Object*

Description

This methods function returns important matrices etc. of a RR-VGLM object.

Usage

```
Coef.rrvglm(object, ...)
```

Arguments

`object`	An object of class `"rrvglm"`.
`...`	Currently unused.

Details

The **A**, **B1**, **C** matrices are returned, along with other slots. See `rrvglm` for details about RR-VGLMs.

Value

An object of class `"Coef.rrvglm"` (see `Coef.rrvglm-class`).

Note

This function is an alternative to `coef.rrvglm`.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

See Also

Coef.rrvglm-class, print.Coef.rrvglm, rrvglm.

Examples

```
# Rank-1 stereotype model of Anderson (1984)
data(pneumo)
n = nrow(pneumo)
pneumo = transform(pneumo, let=log(exposure.time), x1=runif(n), x2=runif(n))
fit = rrvglm(cbind(normal,mild,severe) ~ let + x1 + x2, multinomial, pneumo)
coef(fit, matrix=TRUE)
Coef(fit)
print(Coef(fit), digits=3)
```

Coef.vlm *Extract Model Coefficients for VLM Objects*

Description

Amongst other things, this function applies inverse link functions to the parameters of intercept-only VGLMs.

Usage

```
Coef.vlm(object, ...)
```

Arguments

object	A fitted model.
...	Arguments which may be passed into coef.

Details

Most **VGAM** family functions apply a link function to the parameters, e.g., positive parameter are often have a log link, parameters between 0 and 1 have a logit link. This function can back-transform the parameter estimate to the original scale.

Value

For intercept-only models (e.g., formula is `y ~ 1`) the back-transformed parameter estimates can be returned.

Warning

This function may not work for *all* **VGAM** family functions. You should check your results on some artificial data before applying it to models fitted to real data.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

See Also

`Coef`, `coef`.

Examples

```
set.seed(123)
nn = 1000
y = rbeta(nn, shape1=1, shape2=3)
# parameters are estimated on a log scale
fit = vglm(y ~ 1, betaff, tr=TRUE, crit="c") # intercept-only model
coef(fit, matrix=TRUE) #  log scale
Coef(fit) # On the original scale
```

`Dagum` *The Dagum Distribution*

Description

Density, distribution function, quantile function and random generation for the Dagum distribution with shape parameters `a` and `p`, and scale parameter `scale`.

Usage

```
ddagum(x, a, scale, p.arg)
pdagum(q, a, scale, p.arg)
qdagum(p, a, scale, p.arg)
rdagum(n, a, scale, p.arg)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`a, p.arg`	shape parameters.
`scale`	scale parameter.

Details

See dagum, which is the **VGAM** family function for estimating the parameters by maximum likelihood estimation.

Value

ddagum gives the density, pdagum gives the distribution function, qdagum gives the quantile function, and rdagum generates random deviates.

Note

The Dagum distribution is a special case of the 4-parameter generalized beta II distribution.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

See Also

dagum, genbetaII.

Examples

```
y = rdagum(n=3000, 4, 6, 2)
fit = vglm(y ~ 1, dagum(init.a=2.1), trace=TRUE, crit="c")
coef(fit, mat=TRUE)
Coef(fit)
```

Fisk *The Fisk Distribution*

Description

Density, distribution function, quantile function and random generation for the Fisk distribution with shape parameter a and scale parameter scale.

Usage

```
dfisk(x, a, scale)
pfisk(q, a, scale)
qfisk(p, a, scale)
rfisk(n, a, scale)
```

Arguments

x, q	vector of quantiles.
p	vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
a	shape parameter.
scale	scale parameter.

Details

See fisk, which is the **VGAM** family function for estimating the parameters by maximum likelihood estimation.

Value

dfisk gives the density, pfisk gives the distribution function, qfisk gives the quantile function, and rfisk generates random deviates.

Note

The Fisk distribution is a special case of the 4-parameter generalized beta II distribution.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

See Also

fisk, genbetaII.

Examples

```
y = rfisk(n=1000, 4, 6)
fit = vglm(y ~ 1, fisk, trace=TRUE, crit="c")
coef(fit, mat=TRUE)
Coef(fit)
```

G1G2G3 *The G1G2G3 Blood Group System*

Description

Estimates the three independent parameters of the the G1G2G3 blood group system.

Usage

```
G1G2G3(link = "logit", earg=list(), ip1 = NULL, ip2 = NULL, iF = NULL)
```

Arguments

`link`	Link function applied to `p1`, `p2` and `f`. See `Links` for more choices.
`earg`	List. Extra argument for the link. See `earg` in `Links` for general information.
`ip1, ip2, iF`	Optional initial value for `p1`, `p2` and `f`.

Details

The parameters `p1` and `p2` are probabilities, so that `p3=1-p1-p2` is the third probability. The parameter `f` is the third independent parameter.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Note

The input can be a 6-column matrix of counts, with columns corresponding to `G_1/G_1`, `G_1/G_2`, `G_1/G_3`, `G_2/G_2`, `G_2/G_3`, `G_3/G_3` (in order). Alternatively, the input can be a 6-column matrix of proportions (so each row adds to 1) and the `weights` argument is used to specify the total number of counts for each row.

Author(s)

T. W. Yee

References

Lange, K. (2002) *Mathematical and Statistical Methods for Genetic Analysis*, 2nd ed. New York: Springer-Verlag.

See Also

`AA.Aa.aa`, `AB.Ab.aB.ab`, `AB.Ab.aB.ab2`, `ABO`, `MNSs`.

Examples

```
y = cbind(108, 196, 429, 143, 513, 559)
fit = vglm(y ~ 1, G1G2G3(link=probit), trace=TRUE, crit="coef")
fit = vglm(y ~ 1, G1G2G3(link=logit, ip1=.3, ip2=.3, iF=.02),
           trace=TRUE, crit="coef")
fit = vglm(y ~ 1, G1G2G3(link="identity"), trace=TRUE)
Coef(fit) # Estimated p1, p2 and f
rbind(y, sum(y)*fitted(fit))
sqrt(diag(vcov(fit)))
```

Inv.gaussian *The Inverse Gaussian Distribution*

Description

Density, distribution function and random generation for the inverse Gaussian distribution.

Usage

```
dinv.gaussian(x, mu, lambda)
pinv.gaussian(q, mu, lambda)
rinv.gaussian(n, mu, lambda)
```

Arguments

`x, q`	vector of quantiles.
`n`	number of observations. Must be a single positive integer.
`mu`	the mean parameter.
`lambda`	the λ parameter.

Details

See `inv.gaussianff`, the **VGAM** family function for estimating both parameters by maximum likelihood estimation, for the formula of the probability density function.

Value

`dinv.gaussian` gives the density, `pinv.gaussian` gives the distribution function, and `rinv.gaussian` generates random deviates.

Note

Currently `qinv.gaussian` is unavailable.

Author(s)

T. W. Yee

References

Johnson, N. L. and Kotz, S. and Balakrishnan, N. (1994) *Continuous Univariate Distributions*, 2nd edition, Volume 1, New York: Wiley.

Taraldsen, G. and Lindqvist, B. H. (2005) The multiple roots simulation algorithm, the inverse Gaussian distribution, and the sufficient conditional Monte Carlo method. *Preprint Statistics No. 4/2005*, Norwegian University of Science and Technology, Trondheim, Norway.

See Also

`inv.gaussianff`.

Examples

```
## Not run:
x = seq(-0.05, 4, len=300)
plot(x, dinv.gaussian(x, mu=1, lambda=1), type="l", col="blue", las=1,
     main="blue is density, red is cumulative distribution function")
abline(h=0, col="blue", lty=2)
lines(x, pinv.gaussian(x, mu=1, lambda=1), type="l", col="red")
## End(Not run)
```

Invlomax *The Inverse Lomax Distribution*

Description

Density, distribution function, quantile function and random generation for the inverse Lomax distribution with shape parameter `p` and scale parameter `scale`.

Usage

```
dinvlomax(x, scale, p.arg)
pinvlomax(q, scale, p.arg)
qinvlomax(p, scale, p.arg)
rinvlomax(n, scale, p.arg)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`p.arg`	shape parameter.
`scale`	scale parameter.

Details

See invlomax, which is the **VGAM** family function for estimating the parameters by maximum likelihood estimation.

Value

dinvlomax gives the density, pinvlomax gives the distribution function, qinvlomax gives the quantile function, and rinvlomax generates random deviates.

Note

The inverse Lomax distribution is a special case of the 4-parameter generalized beta II distribution.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

See Also

invlomax, genbetaII.

Examples

```
y = rinvlomax(n=1000, 6, 2)
fit = vglm(y ~ 1, invlomax, trace=TRUE, crit="c")
coef(fit, mat=TRUE)
Coef(fit)
```

Invparalogistic *The Inverse Paralogistic Distribution*

Description

Density, distribution function, quantile function and random generation for the inverse paralogistic distribution with shape parameters a and p, and scale parameter scale.

Usage

```
dinvparalogistic(x, a, scale)
pinvparalogistic(q, a, scale)
qinvparalogistic(p, a, scale)
rinvparalogistic(n, a, scale)
```

Arguments

x, q	vector of quantiles.
p	vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
a	shape parameter.
scale	scale parameter.

Details

See invparalogistic, which is the **VGAM** family function for estimating the parameters by maximum likelihood estimation.

Value

dinvparalogistic gives the density, pinvparalogistic gives the distribution function, qinvparalogistic gives the quantile function, and rinvparalogistic generates random deviates.

Note

The inverse paralogistic distribution is a special case of the 4-parameter generalized beta II distribution.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

See Also

invparalogistic, genbetaII.

Examples

```
y = rinvparalogistic(n=3000, 4, 6)
fit = vglm(y ~ 1, invparalogistic(init.a=2.1), trace=TRUE, crit="c")
coef(fit, mat=TRUE)
Coef(fit)
```

Links *Link functions for VGLM/VGAM/etc. families*

Description

The **VGAM** package provides a number of (parameter) link functions which are described in general here. Collectively, they offer the user considerable flexibility for modelling data.

Usage

```
TypicalVGAMlinkFunction(theta, earg=list(), inverse=FALSE,
                        deriv=0, short=TRUE, tag=FALSE)
```

Arguments

Almost all **VGAM** link functions have something similar to the argument list as given above. That is, there is a matching `earg` for each `link` argument. In the following we have $\eta = g(\theta)$ where g is the link function, θ is the parameter and η is the linear/additive predictor.

`theta`	Numeric or character. Actually this can be θ (default) or η, depending on the other arguments. If `theta` is character then `inverse` and `deriv` are ignored.
`earg`	List. Extra argument allowing for additional information, specific to the link function. For example, for `logoff`, this will contain the offset value. The argument `earg` is always a list with *named* components. See each specific link function to find the component names for the list. Almost all **VGAM** family functions with a single link function have an argument (often called `earg`) which will allow parameters to be inputted for that link function. For **VGAM** family functions with more than one link function there usually will be an `earg`-type argument for each link. For example, if there are two links called `lshape` and `lscale` then the `earg`-type arguments for these might be called `eshape` and `escale`, say.
`inverse`	Logical. If `TRUE` the inverse link value θ is returned, hence the argument `theta` is really η.
`deriv`	Integer. Either 0, 1, or 2 specifying the order of the derivative.
`short, tag`	Logical. Used for labelling the `blurb` slot of a `vglmff-class` object. Used only if `theta` is character, and gives the formula for the link in character form. If `tag=TRUE` then the result contains a little more information.

Details

The following is a brief enumeration of all **VGAM** link functions.

For parameters lying between 0 and 1 (e.g., probabilities): `logit`, `probit`, `cloglog`, `cauchit`, `loglog`, `fsqrt`, `logc`, `golf`, `polf`, `nbolf`.

For positive parameters (i.e., greater than 0): `loge`, `nloge`, `powl`.

For parameters greater than 1: `loglog`.

For parameters between -1 and 1: `fisherz`, `rhobit`.

For parameters between A and B: `elogit`, `logoff` ($B = \infty$).

For unrestricted parameters (i.e., any value): `identity`, `nidentity`, `reciprocal`, `nreciprocal`.

Value

Returns one of the link function value or its first or second derivative, the inverse link or its first or second derivative, or a character description of the link.

Here are the general details. If `inverse=FALSE` and `deriv=0` (default) then the ordinary link function $\eta = g(\theta)$ is returned. If `inverse=FALSE` and `deriv=1` then it is $d\theta/d\eta$ *as a function of* θ. If `inverse=FALSE` and `deriv=2` then it is $d^2\theta/d\eta^2$ *as a function of* θ.

If `inverse=TRUE` and `deriv=0` then the inverse link function is returned, hence `theta` is really η. If `inverse=TRUE` and `deriv` is positive then the *reciprocal* of the same link function with `(theta=theta, earg=earg, inverse=TRUE, deriv=deriv)` is returned.

Note

From October 2006 onwards, all **VGAM** family functions will only contain one default value for each link argument rather than giving a vector of choices. For example, rather than `binomialff(link=c("logit", "probit", "cloglog", "cauchit", "identity"), ...)` it is now `binomialff(link="logit", ...)` No checking will be done to see if the user's choice is reasonable. This means that the user can write his/her own **VGAM** link function and use it within any **VGAM** family function. Altogether this provides greater flexibility. The downside is that the user must specify the *full* name of the link function, by either assigning the link argument the full name as a character string, or just the name itself. See the examples below.

Author(s)

T. W. Yee

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

See Also

`vglm`, `vgam`, `rrvglm`. `cqo`, `cao`, `uqo`.

Examples

```
logit("a")
logit("a", short=FALSE)
logit("a", short=FALSE, tag=TRUE)

logoff(1:5, earg=list(offset=1))  # Same as log(1:5 + 1)
powl(1:5, earg=list(power=2))     # Same as (1:5)^2

data(hunua)
```

```
fit1 = vgam(agaaus ~ altitude, binomialff(link=cloglog), hunua)    # ok
fit2 = vgam(agaaus ~ altitude, binomialff(link="cloglog"), hunua)  # ok

## Not run:
# This no longer works since "clog" is not a valid VGAM link function:
fit3 = vgam(agaaus ~ altitude, binomialff(link="clog"), hunua) # not ok

# No matter what the link, the estimated var-cov matrix is the same
y = rbeta(n=1000, shape1=exp(0), shape2=exp(1))
fit1 = vglm(y ~ 1, betaff(link="identity"), trace = TRUE, crit="c")
fit2 = vglm(y ~ 1, betaff(link=logoff, earg=list(offset=1.1)),
            trace = TRUE, crit="c")
vcov(fit1, untran=TRUE)
vcov(fit1, untran=TRUE)-vcov(fit2, untran=TRUE)  # Should be all 0s
fit1@misc$earg   # No 'special' parameters
fit2@misc$earg   # Some 'special' parameters are here

par(mfrow=c(2,2))
p = seq(0.01, 0.99, len=200)
x = seq(-4, 4, len=200)
plot(p, logit(p), type="l", col="blue")
plot(x, logit(x, inverse=TRUE), type="l", col="blue")
plot(p, logit(p, deriv=1), type="l", col="blue") # reciprocal!
plot(p, logit(p, deriv=2), type="l", col="blue") # reciprocal!
## End(Not run)
```

`Lomax` *The Lomax Distribution*

Description

Density, distribution function, quantile function and random generation for the Lomax distribution with scale parameter `scale` and shape parameter `q`.

Usage

```
dlomax(x, scale, q.arg)
plomax(q, scale, q.arg)
qlomax(p, scale, q.arg)
rlomax(n, scale, q.arg)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`q.arg`	shape parameter.
`scale`	scale parameter.

Details

See lomax, which is the **VGAM** family function for estimating the parameters by maximum likelihood estimation.

Value

dlomax gives the density, plomax gives the distribution function, qlomax gives the quantile function, and rlomax generates random deviates.

Note

The Lomax distribution is a special case of the 4-parameter generalized beta II distribution.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

See Also

lomax, genbetaII.

Examples

```
y = rlomax(n=2000, 6, 2)
fit = vglm(y ~ 1, lomax(init.q=2.1), trace=TRUE, crit="c")
coef(fit, mat=TRUE)
Coef(fit)
```

MNSs *The MNSs Blood Group System*

Description

Estimates the three independent parameters of the the MNSs blood group system.

Usage

```
MNSs(link = "logit", earg=list(), imS = NULL, ims = NULL, inS = NULL)
```

Arguments

link	Link function applied to the three parameters. See Links for more choices.
earg	List. Extra argument applied to each of the links. See earg in Links for general information.
imS, ims, inS	Optional initial value for mS, ms and nS respectively. A NULL means they are computed internally.

Details

There are three independent parameters: m_S, m_s, n_S, say, so that n_s = 1 - m_S - m_s - n_S. We let the eta vector (transposed) be (g(m_S), g(m_s), g(n_S)) where g is the link function.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

The input can be a 6-column matrix of counts, where the columns are MS, Ms, MNS, MNs, NS, Ns (in order). Alternatively, the input can be a 6-column matrix of proportions (so each row adds to 1) and the weights argument is used to specify the total number of counts for each row.

Author(s)

T. W. Yee

References

Elandt-Johnson, R. C. (1971) *Probability Models and Statistical Methods in Genetics*, New York: Wiley.

See Also

AA.Aa.aa, AB.Ab.aB.ab, AB.Ab.aB.ab2, ABO, G1G2G3.

Examples

```
# Order matters only:
y = cbind(MS=295, Ms=107, MNS=379, MNs=322, NS=102, Ns=214)
fit = vglm(y ~ 1, MNSs("logit", .25, .28, .08), trace=TRUE)
fit = vglm(y ~ 1, MNSs(link=logit), trace=TRUE, cri="coef")
Coef(fit)
rbind(y, sum(y)*fitted(fit))
diag(vcov(fit))^0.5
```

`Max` *Maxima*

Description

Generic function for the *maxima* (maximums) of a model.

Usage

```
Max(object, ...)
```

Arguments

`object`	An object for which the computation or extraction of a maximum (or maxima) is meaningful.
`...`	Other arguments fed into the specific methods function of the model. Sometimes they are fed into the methods function for `Coef`.

Details

Different models can define a maximum in different ways. Many models have no such notion or definition.

Maxima occur in quadratic and additive ordination, e.g., CQO or UQO or CAO. For these models the maximum is the fitted value at the optimum. For quadratic ordination models there is a formula for the optimum but for additive ordination models the optimum must be searched for numerically. If it occurs on the boundary, then the optimum is undefined. For a valid optimum, the fitted value at the optimum is the maximum.

Value

The value returned depends specifically on the methods function invoked.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2004) A new technique for maximum-likelihood canonical Gaussian ordination. *Ecological Monographs*, **74**, 685–701.

Yee, T. W. (2006) Constrained additive ordination. *Ecology*, **87**, 203–213.

See Also

`Max.qrrvglm`, `Tol`, `Opt`.

Examples

```
## Not run:
data(hspider)
set.seed(111)  # This leads to the global solution
hspider[,1:6] = scale(hspider[,1:6]) # Standardized environmental vars
p1 = cqo(cbind(Alopacce, Alopcune, Alopfabr, Arctlute, Arctperi,
               Auloalbi, Pardlugu, Pardmont, Pardnigr, Pardpull,
               Trocterr, Zoraspin) ~
         WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + ReflLux,
         Bestof = 2,
         fam = quasipoissonff, data = hspider, Crow1positive=FALSE)

Max(p1)

index = 1:ncol(p1@y)
persp(p1, col=index, las=1, lwd=2)
abline(h=Max(p1), lty=2, col=index)

## End(Not run)
```

Maxwell *The Maxwell Distribution*

Description

Density, and distribution function for the Maxwell distribution.

Usage

```
dmaxwell(x, a)
pmaxwell(q, a)
qmaxwell(p, a)
rmaxwell(n, a)
```

Arguments

x, q	vector of quantiles.
p	vector of probabilities.
n	number of observations. A single positive integer.
a	the parameter.

Details

See maxwell, the **VGAM** family function for estimating the parameter a by maximum likelihood estimation, for the formula of the probability density function.

Value

`dmaxwell` gives the density, `pmaxwell` gives the distribution function, `qmaxwell` gives the quantile function, and `rmaxwell` generates random deviates.

Note

The Maxwell distribution is related to the Rayleigh distribution.

Author(s)

T. W. Yee

References

Balakrishnan, N. and Nevzorov, V. B. (2003) *A Primer on Statistical Distributions*. Hoboken, New Jersey: Wiley.

See Also

`maxwell`, `Rayleigh`, `rayleigh`.

Examples

```
## Not run:
a = 3
x = seq(-0.5, 3, len=100)
plot(x, dmaxwell(x, a=a), type="l", col="blue", las=1, ylab="",
     main="blue is density, red is cumulative distribution function",
     sub="Purple lines are the 10,20,...,90 percentiles")
abline(h=0, col="blue", lty=2)
lines(x, pmaxwell(x, a=a), type="l", col="red")
probs = seq(0.1, 0.9, by=0.1)
Q = qmaxwell(probs, a=a)
lines(Q, dmaxwell(Q, a), col="purple", lty=3, type="h")
lines(Q, pmaxwell(Q, a), col="purple", lty=3, type="h")
abline(h=probs, col="purple", lty=3)
pmaxwell(Q, a) - probs    # Should be all zero
## End(Not run)
```

`Opt` *Maxima*

Description

Generic function for the *optima* (or optimums) of a model.

Usage

```
Opt(object, ...)
```

Arguments

`object`	An object for which the computation or extraction of an optimum (or optima) is meaningful.
`...`	Other arguments fed into the specific methods function of the model. Sometimes they are fed into the methods function for `Coef`.

Details

Different models can define an optimum in different ways. Many models have no such notion or definition.

Optima occur in quadratic and additive ordination, e.g., CQO or UQO or CAO. For these models the optimum is the value of the latent variable where the maximum occurs, i.e., where the fitted value achieves its highest value. For quadratic ordination models there is a formula for the optimum but for additive ordination models the optimum must be searched for numerically. If it occurs on the boundary, then the optimum is undefined. At an optimum, the fitted value of the response is called the *maximum*.

Value

The value returned depends specifically on the methods function invoked.

Note

In ordination, the optimum of a species is sometimes called the *species score*.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2004) A new technique for maximum-likelihood canonical Gaussian ordination. *Ecological Monographs*, **74**, 685–701.

Yee, T. W. (2006) Constrained additive ordination. *Ecology*, **87**, 203–213.

See Also

`Opt.qrrvglm`, `Max`, `Tol`.

Examples

```
data(hspider)
set.seed(111)  # This leads to the global solution
hspider[,1:6] = scale(hspider[,1:6]) # Standardized environmental vars
p1 = cqo(cbind(Alopacce, Alopcune, Alopfabr, Arctlute, Arctperi,
               Auloalbi, Pardlugu, Pardmont, Pardnigr, Pardpull,
               Trocterr, Zoraspin) ~
         WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + ReflLux,
         Bestof = 2,
```

```
          fam = quasipoissonff, data = hspider, Crow1positive=FALSE)
Opt(p1)

## Not run:
index = 1:ncol(p1@y)
persp(p1, col=index, las=1, lwd=2, main="Vertical lines at the optima")
abline(v=Opt(p1), lty=2, col=index)
## End(Not run)
```

`Paralogistic` *The Paralogistic Distribution*

Description

Density, distribution function, quantile function and random generation for the paralogistic distribution with shape parameter `a` and scale parameter `scale`.

Usage

```
dparalogistic(x, a, scale)
pparalogistic(q, a, scale)
qparalogistic(p, a, scale)
rparalogistic(n, a, scale)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`a`	shape parameter.
`scale`	scale parameter.

Details

See `paralogistic`, which is the **VGAM** family function for estimating the parameters by maximum likelihood estimation.

Value

`dparalogistic` gives the density, `pparalogistic` gives the distribution function, `qparalogistic` gives the quantile function, and `rparalogistic` generates random deviates.

Note

The paralogistic distribution is a special case of the 4-parameter generalized beta II distribution.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

See Also

paralogistic, genbetaII.

Examples

```
y = rparalogistic(n=3000, 4, 6)
fit = vglm(y ~ 1, paralogistic(init.a=2.1), trace=TRUE, crit="c")
coef(fit, mat=TRUE)
Coef(fit)
```

Pareto *The Pareto Distribution*

Description

Density, distribution function, quantile function and random generation for the Pareto(I) distribution with parameters `location` and `shape`.

Usage

```
dpareto(x, location, shape)
ppareto(q, location, shape)
qpareto(p, location, shape)
rpareto(n, location, shape)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. Must be a single positive integer.
`location, shape`	the α and k parameters.

Details

See pareto1, the **VGAM** family function for estimating the parameter k by maximum likelihood estimation, for the formula of the probability density function and the range restrictions imposed on the parameters.

Value

`dpareto` gives the density, `ppareto` gives the distribution function, `qpareto` gives the quantile function, and `rpareto` generates random deviates.

Author(s)

T. W. Yee

References

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

See Also

`pareto1`, `ParetoIV`.

Examples

```
alpha = 3; k = exp(1); x = seq(2.8, 8, len=300)
## Not run:
plot(x, dpareto(x, location=alpha, shape=k), type="l",
     main="Pareto density split into 10 equal areas")
abline(h=0, col="blue", lty=2)
qq = qpareto(seq(0.1,0.9,by=0.1),location=alpha,shape=k)
lines(qq, dpareto(qq, loc=alpha, shape=k), col="purple", lty=3, type="h")
## End(Not run)
pp = seq(0.1,0.9,by=0.1)
qq = qpareto(pp, location=alpha, shape=k)
ppareto(qq, location=alpha, shape=k)
qpareto(ppareto(qq,loc=alpha,shape=k),loc=alpha,shape=k) - qq # Should be 0
```

`ParetoIV` *The Pareto(IV/III/II) Distributions*

Description

Density, distribution function, quantile function and random generation for the Pareto(IV/III/II) distributions.

Usage

```
dparetoIV(x, location=0, scale=1, inequality=1, shape=1)
pparetoIV(q, location=0, scale=1, inequality=1, shape=1)
qparetoIV(p, location=0, scale=1, inequality=1, shape=1)
rparetoIV(n, location=0, scale=1, inequality=1, shape=1)
dparetoIII(x, location=0, scale=1, inequality=1)
pparetoIII(q, location=0, scale=1, inequality=1)
```

```
qparetoIII(p, location=0, scale=1, inequality=1)
rparetoIII(n, location=0, scale=1, inequality=1)
dparetoII(x, location=0, scale=1, shape=1)
pparetoII(q, location=0, scale=1, shape=1)
qparetoII(p, location=0, scale=1, shape=1)
rparetoII(n, location=0, scale=1, shape=1)
dparetoI(x, scale=1, shape=1)
pparetoI(q, scale=1, shape=1)
qparetoI(p, scale=1, shape=1)
rparetoI(n, scale=1, shape=1)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. Must be a single positive integer.
`location`	the location parameter.
`scale, shape, inequality`	the (positive) scale, inequality and shape parameters.

Details

For the formulas and other details see `paretoIV`.

Value

Functions beginning with the letter `d` give the density, functions beginning with the letter `p` give the distribution function, functions beginning with the letter `q` give the quantile function, and functions beginning with the letter `r` generates random deviates.

Note

The functions `[dpqr]paretoI` are the same as `[dpqr]pareto1` except for a slight change in notation: $s = k$ and $b = \alpha$; see `Pareto`.

Author(s)

T. W. Yee

References

Brazauskas, V. (2003) Information matrix for Pareto(IV), Burr, and related distributions. *Comm. Statist. Theory and Methods* **32**, 315–325.

Arnold, B. C. (1983) *Pareto Distributions*. Fairland, Maryland: International Cooperative Publishing House.

See Also

`paretoIV`, `Pareto`.

Examples

```
## Not run:
x = seq(-0.2, 4, by=0.01)
loc = 0; Scale = 1; ineq = 1; shape = 1.0;
plot(x, dparetoIV(x, loc, Scale, ineq, shape), type="l", col="blue",
     main="Blue is density, red is cumulative distribution function",
     sub="Purple are 5,10,...,95 percentiles", ylim=0:1, las=1, ylab="")
abline(h=0, col="blue", lty=2)
Q = qparetoIV(seq(0.05,0.95,by=0.05), loc, Scale, ineq, shape)
lines(Q, dparetoIV(Q, loc, Scale, ineq, shape), col="purple", lty=3, type="h")
lines(x, pparetoIV(x, loc, Scale, ineq, shape), col="red")
abline(h=0, lty=2)
## End(Not run)
```

Rayleigh *The Rayleigh Distribution*

Description

Density, distribution function, quantile function and random generation for the Rayleigh distribution with parameter `a`.

Usage

```
drayleigh(x, a)
prayleigh(q, a)
qrayleigh(p, a)
rrayleigh(n, a)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. Must be a positive integer of length 1.
`a`	the parameter a.

Details

See `rayleigh`, the **VGAM** family function for estimating the parameter a by maximum likelihood estimation, for the formula of the probability density function and range restrictions on the parameter a.

Value

`drayleigh` gives the density, `prayleigh` gives the distribution function, `qrayleigh` gives the quantile function, and `rrayleigh` generates random deviates.

Note

The Rayleigh distribution is related to the Maxwell distribution.

Author(s)

T. W. Yee

References

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

See Also

rayleigh, maxwell.

Examples

```
## Not run:
a = 2
x = seq(-1, 8, by=0.1)
plot(x, drayleigh(x, a=a), type="l", ylim=c(0,1), las=1, ylab="",
     main="Rayleigh density divided into 10 equal areas; red=cdf")
abline(h=0, col="blue", lty=2)
qq = qrayleigh(seq(0.1,0.9,by=0.1),a=a)
lines(qq, drayleigh(qq, a=a), col="purple", lty=3, type="h")
lines(x, prayleigh(x, a=a), col="red")
## End(Not run)
```

Sinmad *The Singh-Maddala Distribution*

Description

Density, distribution function, quantile function and random generation for the Singh-Maddala distribution with shape parameters `a` and `q`, and scale parameter `scale`.

Usage

```
dsinmad(x, a, scale, q.arg)
psinmad(q, a, scale, q.arg)
qsinmad(p, a, scale, q.arg)
rsinmad(n, a, scale, q.arg)
```

Arguments

x, q	vector of quantiles.
p	vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
a, q.arg	shape parameters.
scale	scale parameter.

Details

See sinmad, which is the **VGAM** family function for estimating the parameters by maximum likelihood estimation.

Value

dsinmad gives the density, psinmad gives the distribution function, qsinmad gives the quantile function, and rsinmad generates random deviates.

Note

The Singh-Maddala distribution is a special case of the 4-parameter generalized beta II distribution.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

See Also

sinmad, genbetaII.

Examples

```
y = rsinmad(n=3000, 4, 6, 2)
fit = vglm(y ~ 1, sinmad(init.a=2.1), trace=TRUE, crit="c")
coef(fit, mat=TRUE)
Coef(fit)
```

Surv *Create a Survival Object*

Description

Create a survival object, usually used as a response variable in a model formula.

Usage

```
Surv(time, time2, event, type =, origin = 0)
is.SurvS4(x)
```

Arguments

time	for right censored data, this is the follow up time. For interval data, the first argument is the starting time for the interval.
x	any R object.
event	The status indicator, normally 0=alive, 1=dead. Other choices are TRUE/FALSE (TRUE = death) or 1/2 (2=death). For interval censored data, the status indicator is 0=right censored, 1=event at time, 2=left censored, 3=interval censored. Although unusual, the event indicator can be omitted, in which case all subjects are assumed to have an event.
time2	ending time of the interval for interval censored or counting process data only. Intervals are assumed to be open on the left and closed on the right, (start, end]. For counting process data, event indicates whether an event occurred at the end of the interval.
type	character string specifying the type of censoring. Possible values are "right", "left", "counting", "interval", or "interval2". The default is "right" or "counting" depending on whether the time2 argument is absent or present, respectively.
origin	for counting process data, the hazard function origin. This is most often used in conjunction with a model containing time dependent strata in order to align the subjects properly when they cross over from one strata to another.

Details

Typical usages are

```
Surv(time, event)
Surv(time, time2, event, type=, origin=0)
```

In theory it is possible to represent interval censored data without a third column containing the explicit status. Exact, right censored, left censored and interval censored observation would be represented as intervals of (a,a), (a, infinity), (-infinity,b), and (a,b) respectively; each specifying the interval within which the event is known to have occurred.

If `type = "interval2"` then the representation given above is assumed, with NA taking the place of infinity. If 'type="interval" `event` must be given. If `event` is `0`, `1`, or `2`, the relevant information is assumed to be contained in `time`, the value in `time2` is ignored, and the second column of the result will contain a placeholder.

Presently, the only methods allowing interval censored data are the parametric models computed by `survreg`, so the distinction between open and closed intervals is unimportant. The distinction is important for counting process data and the Cox model.

The function tries to distinguish between the use of 0/1 and 1/2 coding for left and right censored data using `if (max(status)==2)`. If 1/2 coding is used and all the subjects are censored, it will guess wrong. Use 0/1 coding in this case.

Value

An object of class `Surv`. There are methods for `print`, `is.na`, and subscripting survival objects. `Surv` objects are implemented as a matrix of 2 or 3 columns.

In the case of `is.SurvS4`, a logical value `TRUE` if `x` inherits from class `"SurvS4"`, otherwise an `FALSE`.

Note

The purpose of having `Surv` in **VGAM** is so that the same input can be fed into `vglm` as functions in **survival** such as `survreg`. The class name has been changed from `"Surv"` to `"SurvS4"`.

Author(s)

The code and documentation comes from **survival**. Slight modifications have been made for conversion to S4 by Thomas W. Yee.

See Also

`SurvS4-class`, `coxph`, `survfit`, `survreg`, `aml`.

Examples

```
with(aml, Surv(time, status))
```

SurvS4-class *Class "SurvS4"*

Description

S4 version of the Surv class.

Objects from the Class

A virtual Class: No objects may be created from it.

Extends

Class "Surv", directly. Class "matrix", directly. Class "oldClass", by class "Surv", distance 2. Class "structure", by class "matrix", distance 2. Class "array", by class "matrix", distance 2. Class "vector", by class "matrix", distance 3, with explicit coerce. Class "vector", by class "matrix", distance 4, with explicit coerce.

Methods

print `signature(x = "SurvS4")`: ...

show `signature(object = "SurvS4")`: ...

Warning

This code has not been thoroughly tested.

Note

The purpose of having `Surv` in **VGAM** is so that the same input can be fed into `vglm` as functions in **survival** such as `survreg`.

Author(s)

Thomas W. Yee.

References

See **survival**.

See Also

`Surv`.

Examples

```
showClass("SurvS4")
```

Tol *Tolerances*

Description

Generic function for the *tolerances* of a model.

Usage

```
Tol(object, ...)
```

Arguments

`object`	An object for which the computation or extraction of a tolerance or tolerances is meaningful.
`...`	Other arguments fed into the specific methods function of the model. Sometimes they are fed into the methods function for `Coef`.

Details

Different models can define an optimum in different ways. Many models have no such notion or definition.

Tolerances occur in quadratic ordination, i.e., CQO or UQO. They have ecological meaning because a high tolerance for a species means the species can survive over a large environmental range (stenoecous species), whereas a small tolerance means the species' niche is small (eurycous species). Mathematically, the tolerance is like the variance of a normal distribution.

Value

The value returned depends specifically on the methods function invoked.

Warning

There is a direct inverse relationship between the scaling of the latent variables (site scores) and the tolerances. One normalization is for the latent variables to have unit variance. Another normalization is for all the tolerances to be unit. These two normalization cannot simultaneously hold in general. For rank-*R*>*1* models it becomes more complicated because the latent variables are also uncorrelated. An important argument when fitting quadratic ordination models is whether `EqualTolerances` is `TRUE` or `FALSE`. See Yee (2004) for details.

Note

Tolerances are undefined for 'linear' and additive ordination models. They are well-defined for quadratic ordination models.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2004) A new technique for maximum-likelihood canonical Gaussian ordination. *Ecological Monographs*, **74**, 685–701.

Yee, T. W. (2006) Constrained additive ordination. *Ecology*, **87**, 203–213.

See Also

`Tol.qrrvglm`. `Max`, `Opt`.

Examples

```
data(hspider)
set.seed(111)  # This leads to the global solution
hspider[,1:6] = scale(hspider[,1:6]) # Standardized environmental vars
p1 = cqo(cbind(Alopacce, Alopcune, Alopfabr, Arctlute, Arctperi,
               Auloalbi, Pardlugu, Pardmont, Pardnigr, Pardpull,
               Trocterr, Zoraspin) ~
         WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + ReflLux,
         Bestof = 2,
         fam = quasipoissonff, data = hspider, Crow1positive=FALSE)

Tol(p1)
```

`VGAM-package` *Vector Generalized Linear and Additive Models*

Description

VGAM provides functions for fitting vector generalized linear and additive models (VGLMs and VGAMs), and associated models (Reduced-Rank VGLMs, Quadratic RR-VGLMs, Reduced-Rank VGAMs). This package fits many models and distributions by maximum likelihood estimation (MLE) or penalized MLE. Also fits constrained ordination models in ecology such as constrained quadratic ordination (CQO).

Details

This package centers on the iteratively reweighted least squares (IRLS) algorithm. Other key words include Fisher scoring, additive models, penalized likelihood, reduced-rank regression and constrained ordination. The central modelling functions are `vglm`, `vgam`, `rrvglm`, `cqo`, `cao`. For detailed control of fitting, each of these has its own control function, e.g., `vglm.control`. The package uses S4 (see `methods-package`).

The classes of GLMs and GAMs are special cases of VGLMs and VGAMs. The VGLM/VGAM framework is intended to be very general so that it encompasses as many distributions and models as possible. VGLMs are limited only by the assumption that the regression coefficients enter through a set of linear predictors. The VGLM class is very large and encompasses a wide range of multivariate response types and models, e.g., it includes univariate and multivariate distributions, categorical data analysis, time series, survival analysis, generalized estimating equations, extreme values, correlated binary data, bioassay data and nonlinear least-squares problems.

VGAMs are to VGLMs what GAMs are to GLMs. Vector smoothing (see `vsmooth.spline`) allows several additive predictors to be estimated as a sum of smooth functions of the covariates.

For a complete list of this package, use `library(help="VGAM")`. New **VGAM** family functions are continually being written and added to the package. A monograph about VGLM and VGAMs etc. is in the making but unfortunately won't be finished for a while.

Author(s)

Thomas W. Yee <t.yee@auckland.ac.nz>

Maintainer: Thomas Yee <t.yee@auckland.ac.nz>

References

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

Yee, T. W. and Stephenson, A. G. (2007) Vector generalized linear and additive extreme value models. *Extremes*, **10**, 1–19.

Yee, T. W. and Wild, C. J. (1996) Vector generalized additive models. *Journal of the Royal Statistical Society, Series B, Methodological*, **58**, 481–493.

Yee, T. W. (2004) A new technique for maximum-likelihood canonical Gaussian ordination. *Ecological Monographs*, **74**, 685–701.

Yee, T. W. (2006) Constrained additive ordination. *Ecology*, **87**, 203–213.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee/VGAM contains further information and examples.

Examples

```
# Example 1
# Fit the proportional odds model, p.179, in McCullagh and Nelder (1989)
data(pneumo)
pneumo = transform(pneumo, let=log(exposure.time))
(fit = vglm(cbind(normal, mild, severe) ~ let,
            cumulative(parallel=TRUE, reverse=TRUE), pneumo))
fit@y   # Sample proportions
weights(fit, type="prior")   # Number of observations
coef(fit, matrix=TRUE)
constraints(fit)   # Constraint matrices

# Example 2
# Zero-inflated Poisson model
x = runif(n <- 2000)
phi = logit(-0.5 + 1*x, inverse=TRUE)
lambda = loge(0.5 + 2*x, inverse=TRUE)
y = rzipois(n, lambda, phi)
table(y)
fit = vglm(y ~ x, zipoisson, trace=TRUE)
coef(fit, matrix=TRUE)  # These should agree with the above values

# Example 3
# Fit a two species GAM simultaneously
data(hunua)
fit2 = vgam(cbind(agaaus, kniexc) ~ s(altitude, df=c(2,3)),
            binomialff(mv=TRUE), hunua)
coef(fit2, mat=TRUE)   # Not really interpretable
## Not run:
plot(fit2, se=TRUE, overlay=TRUE, lcol=1:2, scol=1:2)
attach(hunua)
o = order(altitude)
matplot(altitude[o], fitted(fit2)[o,], type="l", lwd=2, las=1,
    xlab="Altitude (m)", ylab="Probability of presence",
    main="Two plant species' response curves", ylim=c(0,.8))
```

```
rug(altitude)
detach(hunua)
## End(Not run)

# Example 4
# LMS quantile regression
data(bminz)
fit = vgam(BMI ~ s(age, df=c(4,2)), fam=lms.bcn(zero=1), data=bminz, tr=TRUE)
predict(fit)[1:3,]
fitted(fit)[1:3,]
bminz[1:3,]
# Person 1 is near the lower quartile of BMI amongst people his age
cdf(fit)[1:3]

## Not run:
# Quantile plot
par(mfrow=c(1,1), bty="l", mar=c(5,4,4,3)+0.1, xpd=TRUE)
qtplot(fit, percentiles=c(5,50,90,99), main="Quantiles",
       xlim=c(15,90), las=1, ylab="BMI", lwd=2, lcol=4)

# Density plot
ygrid = seq(15, 43, len=100)  # BMI ranges
par(mfrow=c(1,1), lwd=2)
a = deplot(fit, x0=20, y=ygrid, xlab="BMI", col="black",
    main="Density functions at Age = 20 (black), 42 (red) and 55 (blue)")
a
a = deplot(fit, x0=42, y=ygrid, add=TRUE, llty=2, col="red")
a = deplot(fit, x0=55, y=ygrid, add=TRUE, llty=4, col="blue", Attach=TRUE)
a@post$deplot  # Contains density function values
## End(Not run)

# Example 5
# GEV distribution for extremes
data(oxtemp)
(fit = vglm(maxtemp ~ 1, egev, data=oxtemp, trace=TRUE))
fitted(fit)[1:3,]
coef(fit, mat=TRUE)
Coef(fit)
vcov(fit)
vcov(fit, untransform=TRUE)
sqrt(diag(vcov(fit)))   # Approximate standard errors
## Not run:  rlplot(fit)
```

`acat` *Ordinal Regression with Adjacent Categories Probabilities*

Description

Fits an adjacent categories regression model to an ordered (preferably) factor response.

Usage

```
acat(link = "loge", earg = list(),
     parallel = FALSE, reverse = FALSE, zero = NULL)
```

Arguments

In the following, the response Y is assumed to be a factor with ordered values $1, 2, \ldots, M+1$, so that M is the number of linear/additive predictors η_j.

link	Link function applied to the ratios of the adjacent categories probabilities. See Links for more choices.
earg	List. Extra argument for the link function. See earg in Links for general information.
parallel	A logical, or formula specifying which terms have equal/unequal coefficients.
reverse	Logical. By default, the linear/additive predictors used are $\eta_j = \log(P[Y = j+1]/P[Y = j])$ for $j = 1, \ldots, M$. If reverse is TRUE, then $\eta_j = \log(P[Y = j]/P[Y = j+1])$ will be used.
zero	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,...,M}.

Details

By default, the log link is used because the ratio of two probabilities is positive.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Warning

No check is made to verify that the response is ordinal; see ordered.

Note

The response should be either a matrix of counts (with row sums that are all positive), or a factor. In both cases, the y slot returned by vglm/vgam/rrvglm is the matrix of counts.

For a nominal (unordered) factor response, the multinomial logit model (multinomial) is more appropriate.

Here is an example of the usage of the parallel argument. If there are covariates x1, x2 and x3, then parallel = TRUE ~ x1 + x2 -1 and parallel = FALSE ~ x3 are equivalent. This would constrain the regression coefficients for x1 and x2 to be equal; those of the intercepts and x3 would be different.

Author(s)

Thomas W. Yee

References

Agresti, A. (2002) *Categorical Data Analysis*, 2nd ed. New York: Wiley.

Simonoff, J. S. (2003) *Analyzing Categorical Data*, New York: Springer-Verlag.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

cumulative, cratio, sratio, multinomial, pneumo.

Examples

```
data(pneumo)
pneumo = transform(pneumo, let=log(exposure.time))
(fit = vglm(cbind(normal,mild,severe) ~ let, acat, pneumo))
coef(fit, matrix=TRUE)
constraints(fit)
model.matrix(fit)
```

---

alsqreg *Asymmetric Least Squares Quantile Regression*

Description

Quantile regression using asymmetric least squares error loss.

Usage

```
alsqreg(w=1, method.init=1)
```

Arguments

w	Positive constant controlling the percentile. The larger the value the larger the fitted percentile value (the proportion of points below the "w-regression plane"). The default value of unity results in the ordinary least squares (OLS) solution.
method.init	Integer, either 1 or 2 or 3. Initialization method. Choose another value if convergence fails.

Details

This method was proposed by Efron (1991) and full details can be obtained there. Equation numbers below refer to that article. The model is essentially a linear model (see lm), however, the asymmetric squared error loss function for a residual r is r^2 if $r \leq 0$ and wr^2 if $r > 0$. The solution is the set of regression coefficients that minimize the sum of these over the data set, weighted by the weights argument (so that it can contain frequencies). Newton-Raphson estimation is used here.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

The loglikelihood slot currently does not return the log-likelihood but negative the total asymmetric squared error loss (2.5).

Note

On fitting, the extra slot has list components "w" and "percentile". The latter is the percent of observations below the "w-regression plane", which is the fitted values.

One difficulty is finding the w value giving a specified percentile. One solution is to fit the model within a root finding function such as uniroot; see the example below.

For alsqreg objects, methods functions for the generic functions qtplot and cdf have not been written yet.

Author(s)

Thomas W. Yee

References

Efron, B. (1991) Regression percentiles using asymmetric squared error loss. *Statistica Sinica*, **1**, 93–125.

See Also

bminz, lms.bcn and similar variants are alternative methods for quantile regression.

Examples

```
# Example 1
data(bminz)
o = with(bminz, order(age))
bminz = bminz[o,]  # Sort by age
fit = vglm(BMI ~ bs(age), fam=alsqreg(w=0.07), data=bminz)
fit # Note "loglikelihood" is -total asymmetric squared error loss (2.5)
fit@extra  # Gives the w value and the percentile
coef(fit)
coef(fit, matrix=TRUE)

## Not run:
# Quantile plot
with(bminz, plot(age, BMI, col="blue", main=
     paste(round(fit@extra$percentile, dig=1), "percentile curve")))
with(bminz, lines(age, c(fitted(fit)), col="red"))
## End(Not run)
```

```
# Example 2
# Find the w values that give the 25, 50 and 75 percentiles
findw = function(w, percentile=50) {
    fit = vglm(BMI ~ bs(age), fam=alsqreg(w=w), data=bminz)
    fit@extra$percentile - percentile
}
## Not run:
# Quantile plot
with(bminz, plot(age, BMI, col="blue", las=1, main=
     "25, 50 and 75 percentile curves"))
## End(Not run)
for(myp in c(25,50,75)) {
    bestw = uniroot(f=findw, interval=c(1/10^4, 10^4), percentile=myp)
    fit = vglm(BMI ~ bs(age), fam=alsqreg(w=bestw$root), data=bminz)
## Not run:
    with(bminz, lines(age, c(fitted(fit)), col="red"))
## End(Not run)
}
```

`aml` *Acute Myelogenous Leukemia Survival Data*

Description

Survival in patients with Acute Myelogenous Leukemia

Usage

```
aml
leukemia
```

Format

time:	survival or censoring time
status:	censoring status
x:	maintenance chemotherapy given? (factor)

Note

This data set has been transferred from **survival**.

Source

Rupert G. Miller (1997), *Survival Analysis*. John Wiley & Sons. ISBN: 0-471-25218-2.

auuc *Auckland University Undergraduate Counts*

Description

Undergraduate student enrolments at the University of Auckland in 1990.

Usage

```
data(auuc)
```

Format

A data frame with 4 observations on the following 5 variables.

Commerce a numeric vector of counts.

Arts a numeric vector of counts.

SciEng a numeric vector of counts.

Law a numeric vector of counts.

Medicine a numeric vector of counts.

Details

Each student is cross-classified by their colleges (Science and Engineering have been combined) and the socio-economic status (SES) of their fathers (1 = highest, down to 4 = lowest).

Source

Dr Tony Morrison.

References

Wild, C. J. and Seber, G. A. F. (2000) *Chance Encounters: A First Course in Data Analysis and Inference*, New York: Wiley.

Examples

```
data(auuc)
round(fitted(grc(auuc)))
round(fitted(grc(auuc, Rank=2)))
```

benini *Benini Distribution Family Function*

Description

Estimating the parameter of the Benini distribution by maximum likelihood estimation.

Usage

```
benini(y0=stop("argument \"y0\" must be specified"),
       lshape="loge", earg=list(), ishape=NULL, method.init=1)
```

Arguments

y0	Positive scale parameter.
lshape	Parameter link function applied to the parameter b, which is the shape parameter. See Links for more choices. A log link is the default because b is positive.
earg	List. Extra argument for the link. See earg in Links for general information.
ishape	Optional initial value for the shape parameter. The default is to compute the value internally.
method.init	An integer with value 1 or 2 which specifies the initialization method. If failure to converge occurs try the other value, or else specify a value for ishape.

Details

The Benini distribution has a probability density function that can be written

$$f(y) = 2b \exp(-b[(\log(y/y_0))^2]) \log(y/y_0)/y$$

for $y_0 > 0$, $y_0 < y$, and $b > 0$. The cumulative distribution function for Y is

$$F(y) = 1 - \exp(-b[(\log(y/y_0))^2]).$$

Here, Newton-Raphson and Fisher scoring coincide.

On fitting, the extra slot has a component called y0 which contains the value of the y0 argument.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Warning

The mean of Y, which are returned as the fitted values, may be incorrect.

Note

Yet to do: the 2-parameter Benini distribution estimates y_0 as well, and the 3-parameter Benini distribution estimates another shape parameter a too.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

See Also

Benini.

Examples

```
y  = rbenini(n <- 3000, y0=1, shape=exp(2))
fit = vglm(y ~ 1, benini(y0=1), trace=TRUE, crit="c")
coef(fit, matrix=TRUE)
Coef(fit)
fit@extra$y0

# Apparent discrepancy:
fitted(fit)[1:5]
mean(y)
```

Benini *The Benini Distribution*

Description

Density, distribution function, quantile function and random generation for the Benini distribution with parameter shape.

Usage

```
dbenini(x, shape, y0)
pbenini(q, shape, y0)
qbenini(p, shape, y0)
rbenini(n, shape, y0)
```

Arguments

x, q	vector of quantiles.
p	vector of probabilities.
n	number of observations. Must be a positive integer of length 1.
shape	the shape parameter b.
y0	the scale parameter y_0.

Details

See benini, the **VGAM** family function for estimating the parameter b by maximum likelihood estimation, for the formula of the probability density function and other details.

Value

dbenini gives the density, pbenini gives the distribution function, qbenini gives the quantile function, and rbenini generates random deviates.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

See Also

benini.

Examples

```
## Not run:
y0 = 1
shape = exp(1)
x = seq(-0.0, 4, len=101)
plot(x, dbenini(x, y0=y0,shape=shape), type="l", col="blue", las=1,
     main="Blue is density, red is cumulative distribution function",
     sub="Purple lines are the 10,20,...,90 percentiles", ylim=0:1,
     ylab="")
abline(h=0, col="blue", lty=2)
lines(x, pbenini(x, y0=y0,shape=shape), col="red")
probs = seq(0.1, 0.9, by=0.1)
Q = qbenini(probs, y0=y0,shape=shape)
lines(Q, dbenini(Q, y0=y0,shape=shape), col="purple", lty=3, type="h")
pbenini(Q, y0=y0,shape=shape) - probs    # Should be all zero
## End(Not run)
```

betaII *Beta Distribution of the Second Kind*

Description

Maximum likelihood estimation of the 3-parameter beta II distribution.

Usage

```
betaII(link.scale = "loge", link.p = "loge", link.q = "loge",
       earg.scale=list(), earg.p=list(), earg.q=list(),
       init.scale = NULL, init.p = 1, init.q = 1, zero = NULL)
```

Arguments

`link.scale, link.p, link.q`
: Parameter link functions applied to the (positive) parameters `scale`, `p` and `q`. See `Links` for more choices.

`earg.scale, earg.p, earg.q`
: List. Extra argument for each of the links. See `earg` in `Links` for general information.

`init.scale, init.p, init.q`
: Optional initial values for `scale`, `p` and `q`.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. Here, the values must be from the set {1,2,3} which correspond to `scale`, `p`, `q`, respectively.

Details

The 3-parameter beta II is the 4-parameter *generalized* beta II distribution with shape parameter $a = 1$. It is also known as the Pearson VI distribution. Other distributions which are special cases of the 3-parameter beta II include the Lomax ($p = 1$) and inverse Lomax ($q = 1$). More details can be found in Kleiber and Kotz (2003).

The beta II distribution has density

$$f(y) = y^{p-1}/[b^p B(p,q)\{1 + y/b\}^{p+q}]$$

for $b > 0$, $p > 0$, $q > 0$, $y > 0$. Here, b is the scale parameter `scale`, and the others are shape parameters. The mean is

$$E(Y) = b\,\Gamma(p+1)\,\Gamma(q-1)/(\Gamma(p)\,\Gamma(q))$$

provided $q > 1$.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

If the self-starting initial values fail, try experimenting with the initial value arguments, especially those whose default value is not NULL.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

See Also

betaff, genbetaII, dagum, sinmad, fisk, invlomax, lomax, paralogistic, invparalogistic.

Examples

```
y = rsinmad(n=2000, a=1, 6, 2)  # Not genuine data!
fit = vglm(y ~ 1, betaII, trace=TRUE)
fit = vglm(y ~ 1, betaII(init.p=0.7, init.q=0.7), trace=TRUE, crit="c")
coef(fit, mat=TRUE)
Coef(fit)
summary(fit)
```

---

betabin.ab *Beta-binomial Distribution Family Function*

Description

Fits a beta-binomial distribution by maximum likelihood estimation. The two parameters here are the shape parameters of the underlying beta distribution.

Usage

```
betabin.ab(link.shape12 = "loge", earg = list(),
           i1 = 1, i2 = NULL, zero = NULL)
```

Arguments

link.shape12	Link function applied to both (positive) shape parameters of the beta distribution. See Links for more choices.
earg	List. Extra argument for the link. See earg in Links for general information.
i1, i2	Initial value for the shape parameters. The first must be positive, and is recyled to the necessary length. The second is optional. If a failure to converge occurs, try assigning a different value to i1 and/or using i2.

`zero` An integer specifying which linear/additive predictor is to be modelled as an intercept only. If assigned, the single value should be either `1` or `2`. The default is to model both shape parameters as functions of the covariates. If a failure to converge occurs, try `zero=2`.

Details

There are several parameterizations of the beta-binomial distribution. This family function directly models the two shape parameters of the associated beta distribution rather than the probability of success (however, see **Note** below). The model can be written $T|P = p \sim Binomial(N, p)$ where P has a beta distribution with shape parameters α and β. Here, N is the number of trials (e.g., litter size), $T = NY$ is the number of successes, and p is the probability of a success (e.g., a malformation). That is, Y is the *proportion* of successes. Like `binomialff`, the fitted values are the estimated probability of success (i.e., $E[Y]$ and not $E[T]$) and the prior weights N are attached separately on the object in a slot.

The probability function is

$$P(T = t) = \binom{N}{t} \frac{B(\alpha + t, \beta + N - t)}{B(\alpha, \beta)}$$

where $t = 0, 1, \ldots, N$, and B is the beta function with shape parameters α and β. Recall $Y = T/N$ is the real response being modelled.

The default model is $\eta_1 = \log(\alpha)$ and $\eta_2 = \log(\beta)$ because both parameters are positive. The mean (of Y) is $p = \mu = \alpha/(\alpha + \beta)$ and the variance (of Y) is $\mu(1 - \mu)(1 + (N - 1)\rho)/N$. Here, the correlation ρ is given by $1/(1 + \alpha + \beta)$ and is the correlation between the N individuals within a litter. A *litter effect* is typically reflected by a positive value of ρ. It is known as the *over-dispersion parameter*.

This family function uses Fisher scoring. The two diagonal elements of the second-order expected derivatives with respect to α and β are computed numerically, which may fail for large α, β, N or else take a long time.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`.

Suppose `fit` is a fitted beta-binomial model. Then `fit@y` contains the sample proportions y, `fitted(fit)` returns estimates of $E(Y)$, and `weights(fit, type="prior")` returns the number of trials N.

Warning

This family function is prone to numerical difficulties due to the expected information matrices not being positive-definite or ill-conditioned over some regions of the parameter space. If problems occur try setting `i1` to be some other positive value, using `i2` and/or setting `zero=2`.

This family function may be renamed in the future.

Note

This function processes the input in the same way as `binomialff`. But it does not handle the case $N = 1$ very well because there are two parameters to estimate, not one, for each row of the input. Cases where $N = 1$ can be omitted via the `subset` argument of `vglm`.

Although the two linear/additive predictors given above are in terms of α and β, basic algebra shows that the default amounts to fitting a logit link to the probability of success; subtracting the second linear/additive predictor from the first gives that logistic regression linear/additive predictor. That is, $logit(p) = \eta_1 - \eta_2$. This is illustated in one of the examples below.

The *extended* beta-binomial distribution of Prentice (1986) is currently not implemented in the **VGAM** package as it has range-restrictions for the correlation parameter that are currently too difficult to handle in this package.

Author(s)

T. W. Yee

References

Moore, D. F. and Tsiatis, A. (1991) Robust estimation of the variance in moment methods for extra-binomial and extra-Poisson variation. *Biometrics*, **47**, 383–401.

Prentice, R. L. (1986) Binary regression using an extended beta-binomial distribution, with discussion of correlation induced by covariate measurement errors. *Journal of the American Statistical Association*, **81**, 321–327.

See Also

`betabinomial`, `Betabin`, `binomialff`, `betaff`, `dirmultinomial`, `lirat`.

Examples

```
# Example 1
N = 10; s1=exp(1); s2=exp(2)
y = rbetabin.ab(n=100, size=N, shape1=s1, shape2=s2)
fit = vglm(cbind(y,N-y) ~ 1, betabin.ab, trace=TRUE)
coef(fit, matrix=TRUE)
Coef(fit)
fit@misc$rho[1:4] # The correlation parameter
cbind(fit@y, weights(fit, type="prior"))[1:5,]

# Example 2
data(lirat)
fit = vglm(cbind(R,N-R) ~ 1, betabin.ab, data=lirat,
           trace=TRUE, subset=N>1)
coef(fit, matrix=TRUE)
Coef(fit)
fit@misc$rho      # The correlation parameter
t(fitted(fit))
t(fit@y)
t(weights(fit, type="prior"))
```

```
# A loge link for the 2 shape parameters results in
# a logistic regression:
all.equal(c(fitted(fit)),
c(logit(predict(fit)[,1] - predict(fit)[,2], inverse=TRUE)))

# Example 3, which is more complicated
lirat = transform(lirat, fgrp = factor(grp))
summary(lirat)   # Only 5 litters in group 3
fit2 = vglm(cbind(R,N-R) ~ fgrp + hb, betabin.ab(zero=2),
           data=lirat, trace=TRUE, subset=N>1)
coef(fit2, matrix=TRUE)
Coef(fit2)
coef(fit2, matrix=TRUE)[,1] - coef(fit2, matrix=TRUE)[,2] # logit(p)
## Not run:
plot(lirat$hb[lirat$N>1], fit2@misc$rho,
     xlab="Hemoglobin", ylab="Estimated rho",
     pch=as.character(lirat$grp[lirat$N>1]),
     col=lirat$grp[lirat$N>1])
## End(Not run)
## Not run:
data(lirat)
attach(lirat)
# cf. Figure 3 of Moore and Tsiatis (1991)
plot(hb, R/N, pch=as.character(grp), col=grp, las=1,
     xlab="Hemoglobin level", ylab="Proportion Dead",
     main="Fitted values (lines)")
detach(lirat)

smalldf = lirat[lirat$N>1,]
for(gp in 1:4) {
    xx = smalldf$hb[smalldf$grp==gp]
    yy = fitted(fit2)[smalldf$grp==gp]
    o = order(xx)
    lines(xx[o], yy[o], col=gp)
}
## End(Not run)
```

`Betabin` *The Beta-Binomial Distribution*

Description

Density, distribution function, and random generation for the beta-binomial distribution.

Usage

```
dbetabin(x, size, prob, rho, log=FALSE)
pbetabin(q, size, prob, rho, log.p=FALSE)
rbetabin(n, size, prob, rho)
dbetabin.ab(x, size, shape1, shape2, log=FALSE)
```

```
pbetabin.ab(q, size, shape1, shape2, log.p=FALSE)
rbetabin.ab(n, size, shape1, shape2)
```

Arguments

x, q	vector of quantiles.
size	number of trials.
n	number of observations. Must be a positive integer of length 1.
prob	the probability of success μ. Must be in the unit open interval $(0, 1)$.
rho	the correlation parameter ρ. Must be in the unit open interval $(0, 1)$.
shape1, shape2	the two (positive) shape parameters of the standard beta distribution. They are called a and b in beta respectively.
log, log.p	Logical. If TRUE then all probabilities p are given as log(p).

Details

The beta-binomial distribution is a binomial distribution whose probability of success is not a constant but it is generated from a beta distribution with parameters shape1 and shape2. Note that the mean of this beta distribution is mu=shape1/(shape1+shape2), which therefore is the mean or the probability of success.

See betabinomial and betabin.ab, the **VGAM** family functions for estimating the parameters, for the formula of the probability density function and other details.

Value

dbetabin and dbetabin.ab give the density, pbetabin and pbetabin.ab give the distribution function, and rbetabin and rbetabin.ab generate random deviates.

Note

pbetabin and pbetabin.ab can be particularly slow. The functions here ending in .ab are called from those functions which don't. The simple transformations $\mu = \alpha/(\alpha + \beta)$ and $\rho = 1/(1 + \alpha + \beta)$ are used, where α and β are the two shape parameters.

Author(s)

T. W. Yee

See Also

betabinomial, betabin.ab.

Examples

```
## Not run:
N = 9; x = 0:N; s1=2; s2=3
dy = dbetabin.ab(x, size=N, shape1=s1, shape2=s2)
plot(x, dy, type="h", col="red", ylim=c(0,0.25), ylab="Probability",
     main=paste("Beta-binomial (size=",N,", shape1=",s1,
                ", shape2=",s2,")", sep=""))
lines(x+0.1, dbinom(x, size=N, prob=s1/(s1+s2)), type="h", col="blue")
sum(dy*x) # Check expected values are equal
sum(dbinom(x, size=N, prob=s1/(s1+s2))*x)
cumsum(dy) -  pbetabin.ab(x, N, shape1=s1, shape2=s2)

y = rbetabin.ab(n=10000, size=N, shape1=s1, shape2=s2)
ty = table(y)
lines(as.numeric(names(ty))+0.2, ty/sum(ty), type="h", col="green")
legend(5, 0.25, leg=c("beta-binomial","binomial", "random generated"),
       col=c("red","blue","green"), lty=1)
## End(Not run)
```

---

betabinomial *Beta-binomial Distribution Family Function*

Description

Fits a beta-binomial distribution by maximum likelihood estimation. The two parameters here are the mean and correlation coefficient.

Usage

```
betabinomial(lmu="logit", lrho="logit", emu=list(), erho=list(),
             irho=NULL, method.init=1, zero=2)
```

Arguments

`lmu, lrho`	Link functions applied to the two parameters. See `Links` for more choices. The defaults ensure the parameters remain in $(0, 1)$.
`emu, erho`	List. Extra argument for each of the links. See `earg` in `Links` for general information.
`irho`	Optional initial value for the correlation parameter. If given, it must be in $(0, 1)$, and is recyled to the necessary length. Assign this argument a value if a convergence failure occurs. Having `irho=NULL` means an initial value is obtained internally, though this can give unsatisfactory results.
`method.init`	An integer with value `1` or `2` which specifies the initialization method for μ. If failure to converge occurs try the other value and/or else specify a value for `irho`.

`zero` An integer specifying which linear/additive predictor is to be modelled as an intercept only. If assigned, the single value should be either `1` or `2`. The default is to have a single correlation parameter. To model both parameters as functions of the covariates assign `zero=NULL`.

Details

There are several parameterizations of the beta-binomial distribution. This family function directly models the mean and correlation parameter, i.e., the probability of success. The model can be written $T|P = p \sim Binomial(N, p)$ where P has a beta distribution with shape parameters α and β. Here, N is the number of trials (e.g., litter size), $T = NY$ is the number of successes, and p is the probability of a success (e.g., a malformation). That is, Y is the *proportion* of successes. Like `binomialff`, the fitted values are the estimated probability of success (i.e., $E[Y]$ and not $E[T]$) and the prior weights N are attached separately on the object in a slot.

The probability function is

$$P(T = t) = \binom{N}{t} \frac{B(\alpha + t, \beta + N - t)}{B(\alpha, \beta)}$$

where $t = 0, 1, \ldots, N$, and B is the beta function with shape parameters α and β. Recall $Y = T/N$ is the real response being modelled.

The default model is $\eta_1 = logit(\mu)$ and $\eta_2 = logit(\rho)$ because both parameters lie between 0 and 1. The mean (of Y) is $p = \mu = \alpha/(\alpha+\beta)$ and the variance (of Y) is $\mu(1-\mu)(1+(N-1)\rho)/N$. Here, the correlation ρ is given by $1/(1+\alpha+\beta)$ and is the correlation between the N individuals within a litter. A *litter effect* is typically reflected by a positive value of ρ. It is known as the *over-dispersion parameter*.

This family function uses Fisher scoring. Elements of the second-order expected derivatives with respect to α and β are computed numerically, which may fail for large α, β, N or else take a long time.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`.

Suppose `fit` is a fitted beta-binomial model. Then `fit@y` contains the sample proportions y, `fitted(fit)` returns estimates of $E(Y)$, and `weights(fit, type="prior")` returns the number of trials N.

Warning

This family function is prone to numerical difficulties due to the expected information matrices not being positive-definite or ill-conditioned over some regions of the parameter space. If problems occur try setting `irho` to some numerical value, or else use `etastart` argument of `vglm`, etc.

Note

This function processes the input in the same way as `binomialff`. But it does not handle the case $N = 1$ very well because there are two parameters to estimate, not one, for each row of the input. Cases where $N = 1$ can be omitted via the `subset` argument of `vglm`.

The *extended* beta-binomial distribution of Prentice (1986) is currently not implemented in the **VGAM** package as it has range-restrictions for the correlation parameter that are currently too difficult to handle in this package.

Author(s)

T. W. Yee

References

Moore, D. F. and Tsiatis, A. (1991) Robust estimation of the variance in moment methods for extra-binomial and extra-Poisson variation. *Biometrics*, **47**, 383–401.

Prentice, R. L. (1986) Binary regression using an extended beta-binomial distribution, with discussion of correlation induced by covariate measurement errors. *Journal of the American Statistical Association*, **81**, 321–327.

See Also

betabin.ab, Betabin, binomialff, betaff, dirmultinomial, lirat.

Examples

```
# Example 1
N = 10; mu = 0.5; rho = 0.8
y = rbetabin(n=100, size=N, prob=mu, rho=rho)
fit = vglm(cbind(y,N-y) ~ 1, betabinomial, trace=TRUE)
coef(fit, matrix=TRUE)
Coef(fit)
cbind(fit@y, weights(fit, type="prior"))[1:5,]

# Example 2
data(lirat)
fit = vglm(cbind(R,N-R) ~ 1, betabinomial, data=lirat,
           trace=TRUE, subset=N>1)
coef(fit, matrix=TRUE)
Coef(fit)
t(fitted(fit))
t(fit@y)
t(weights(fit, type="prior"))

# Example 3, which is more complicated
lirat = transform(lirat, fgrp = factor(grp))
summary(lirat)   # Only 5 litters in group 3
fit2 = vglm(cbind(R,N-R) ~ fgrp + hb, betabinomial(zero=2),
           data=lirat, trace=TRUE, subset=N>1)
coef(fit2, matrix=TRUE)
## Not run:
plot(lirat$hb[lirat$N>1], fit2@misc$rho,
     xlab="Hemoglobin", ylab="Estimated rho",
     pch=as.character(lirat$grp[lirat$N>1]),
     col=lirat$grp[lirat$N>1])
## End(Not run)
```

```
## Not run:
data(lirat)
attach(lirat)
# cf. Figure 3 of Moore and Tsiatis (1991)
plot(hb, R/N, pch=as.character(grp), col=grp, las=1,
     xlab="Hemoglobin level", ylab="Proportion Dead",
     main="Fitted values (lines)")
detach(lirat)

smalldf = lirat[lirat$N>1,]
for(gp in 1:4) {
    xx = smalldf$hb[smalldf$grp==gp]
    yy = fitted(fit2)[smalldf$grp==gp]
    o = order(xx)
    lines(xx[o], yy[o], col=gp)
}
## End(Not run)
```

`betaff` *The Two-parameter Beta Distribution Family Function*

Description

Estimation of the shape parameters of the two-parameter Beta distribution.

Usage

```
betaff(link = "loge", earg=list(),
       i1 = NULL, i2 = NULL, trim = 0.05,
       A = 0, B = 1, zero = NULL)
```

Arguments

`link`	Parameter link function applied to the two shape parameters. See `Links` for more choices. A log link (default) ensures that the parameters are positive.
`earg`	List. Extra argument for the link. See `earg` in `Links` for general information.
`i1, i2`	Initial value for the first and second shape parameters respectively. A `NULL` value means it is obtained in the `initialize` slot.
`trim`	An argument which is fed into `mean()`; it is the fraction (0 to 0.5) of observations to be trimmed from each end of the response `y` before the mean is computed. This is used when computing initial values, and guards against outliers.
`A, B`	Lower and upper limits of the distribution. The defaults correspond to the *standard beta distribution* where the response lies between 0 and 1.
`zero`	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. If used, the value must be from the set {1,2} which correspond to the first and second shape parameters respectively.

Details

The two-parameter Beta distribution is given by $f(y) =$

$$(y-A)^{shape1-1} \times (B-y)^{shape2-1} / [Beta(shape1, shape2) \times (B-A)^{shape1+shape2-1}]$$

for $A < y < B$, and $Beta(.,.)$ is the beta function (see `beta`). The shape parameters are positive, and here, the limits A and B are known. The mean of Y is $E(Y) = A + (B-A) \times shape1/(shape1 + shape2)$, and these are the fitted values of the object.

For the standard beta distribution the variance of Y is $shape1 \times shape2/[(1+shape1+shape2) \times (shape1 + shape2)^2]$. If $\sigma^2 = 1/(1 + shape1 + shape2)$ then the variance of Y can be written $\sigma^2 \mu(1-\mu)$ where $\mu = shape1/(shape1 + shape2)$ is the mean of Y.

If A and B are unknown, then the **VGAM** family function `beta4()` can be used to estimate these too.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

Note

The response must have values in the interval (A, B).

Author(s)

Thomas W. Yee

References

Johnson, N. L. and Kotz, S. and Balakrishnan, N. (1995) Chapter 25 of: *Continuous Univariate Distributions*, 2nd edition, Volume 2, New York: Wiley.

Gupta, A. K. and Nadarajah, S. (2004) *Handbook of Beta Distribution and Its Applications*, NY: Marcel Dekker, Inc.

Documentation accompanying the **VGAM** package at `http://www.stat.auckland.ac.nz/~yee` contains further information and examples.

See Also

`Beta`, `genbetaII`, `betaII`, `betabin.ab`, `betageometric`, `betaprime`, `rbetageom`, `rbetanorm`, `beta4`.

Examples

```
y = rbeta(n=1000, shape1=exp(0), shape2=exp(1))
fit = vglm(y ~ 1, betaff(link="identity"), trace = TRUE, crit="c")
fit = vglm(y ~ 1, betaff, trace = TRUE, crit="c")
coef(fit, matrix=TRUE)
Coef(fit)  # Useful for intercept-only models
```

```
Y = 5 + 8 * y    # From 5 to 13, not 0 to 1
fit = vglm(Y ~ 1, betaff(A=5, B=13), trace = TRUE)
Coef(fit)
fitted(fit)[1:4,]
```

Betageom *The Beta-Geometric Distribution*

Description

Density, distribution function, and random generation for the beta-geometric distribution.

Usage

```
dbetageom(x, shape1, shape2, log=FALSE)
pbetageom(q, shape1, shape2, log.p=FALSE)
rbetageom(n, shape1, shape2)
```

Arguments

x, q	vector of quantiles.
n	number of observations. Must be a positive integer of length 1.
shape1, shape2	the two (positive) shape parameters of the standard beta distribution. They are called a and b in beta respectively.
log, log.p	Logical. If TRUE then all probabilities p are given as log(p).

Details

The beta-geometric distribution is a geometric distribution whose probability of success is not a constant but it is generated from a beta distribution with parameters shape1 and shape2. Note that the mean of this beta distribution is shape1/(shape1+shape2), which therefore is the mean of the probability of success.

Value

dbetageom gives the density, pbetageom gives the distribution function, and rbetageom generates random deviates.

Note

pbetageom can be particularly slow.

Author(s)

T. W. Yee

See Also

geometric, betaff, Beta.

Examples

```
## Not run:
shape1 = 1; shape2 = 2; y = 0:30
proby = dbetageom(y, shape1, shape2, log=FALSE)
plot(y, proby, type="h", col="blue", ylab="P[Y=y]",
     main=paste("Y ~ Beta-geometric(shape1=",shape1,", shape2=",shape2,")",
     sep=""))
sum(proby)
## End(Not run)
```

---

betageometric *Beta-geometric Distribution Family Function*

Description

Maximum likelihood estimation for the beta-geometric distribution.

Usage

```
betageometric(lprob="logit", lshape="loge",
              eprob=list(), eshape=list(),
              iprob = NULL, ishape = 0.1,
              moreSummation=c(2,100), tolerance=1.0e-10, zero=NULL)
```

Arguments

lprob, lshape
: Parameter link functions applied to the parameters p and ϕ (called prob and shape below). The former lies in the unit interval and the latter is positive. See Links for more choices.

eprob, eshape
: List. Extra argument for each of the links. See earg in Links for general information.

iprob, ishape
: Numeric. Initial values for the two parameters. A NULL means a value is computed internally.

moreSummation
: Integer, of length 2. When computing the expected information matrix a series summation from 0 to moreSummation[1]*max(y)+moreSummation[2] is made, in which the upper limit is an approximation to infinity. Here, y is the response.

tolerance
: Positive numeric. When all terms are less than this then the series is deemed to have converged.

zero
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. If used, the value must be from the set {1,2}.

Details

A random variable Y has a 2-parameter beta-geometric distribution if $P(Y = y) = p(1-p)^y$ for $y = 0, 1, 2, \ldots$ where p are generated from a standard beta distribution with shape parameters `shape1` and `shape2`. The parameterization here is to focus on the parameters p and $\phi = 1/(shape1 + shape2)$, where ϕ is `shape`. The default link functions for these ensure that the appropriate range of the parameters is maintained. The mean of Y is $E(Y) = shape2/(shape1-1) = (1-p)/(p-\phi)$.

The geometric distribution is a special case of the beta-geometric distribution with $\phi = 0$ (see `geometric`). However, fitting data from a geometric distribution may result in numerical problems because the estimate of $\log(\phi)$ will 'converge' to `-Inf`.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

The first iteration may be very slow; if practical, it is best for the `weights` argument of `vglm` etc. to be used rather than inputting a very long vector as the response, i.e., `vglm(y ~ 1, ..., weights=wts)` is to be preferred over `vglm(rep(y, wts) ~ 1, ...)`. If convergence problems occur try inputting some values of argument `ishape`.

If an intercept-only model is fitted then the `misc` slot of the fitted object has list components `shape1` and `shape2`.

Author(s)

T. W. Yee

References

Paul, S. R. (2005) Testing goodness of fit of the geometric distribution: an application to human fecundability data. *Journal of Modern Applied Statistical Methods*, **4**, 425–433.

See Also

`geometric`, `betaff`, `rbetageom`.

Examples

```
y = 0:11; wts = c(227,123,72,42,21,31,11,14,6,4,7,28)
fit  = vglm(y ~ 1, fam=betageometric, weight=wts, trace=TRUE)
fitg = vglm(y ~ 1, fam=    geometric, weight=wts, trace=TRUE)
coef(fit, matrix=TRUE)
Coef(fit)
diag(vcov(fit, untrans=TRUE))^0.5
fit@misc$shape1
fit@misc$shape2
# Very strong evidence of a beta-geometric:
1-pchisq(2*(logLik(fit)-logLik(fitg)), df=1)
```

Betanorm *The Beta-Normal Distribution*

Description

Density, distribution function, quantile function and random generation for the univariate beta-normal distribution.

Usage

```
dbetanorm(x, shape1, shape2, mean=0, sd=1, log.arg=FALSE)
pbetanorm(q, shape1, shape2, mean=0, sd=1, lower.tail=TRUE, log.p=FALSE)
qbetanorm(p, shape1, shape2, mean=0, sd=1)
rbetanorm(n, shape1, shape2, mean=0, sd=1)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. Must be a positive integer of length 1.
`shape1, shape2`	the two (positive) shape parameters of the standard beta distribution. They are called `a` and `b` in `beta`.
`mean, sd`	the mean and standard deviation of the univariate normal distribution.
`log.arg, log.p`	Logical. If `TRUE` then all probabilities `p` are given as `log(p)`.
`lower.tail`	Logical. If `TRUE` then the upper tail is returned, i.e., one minus the usual answer.

Details

The function `betanormal1`, the **VGAM** family function for estimating the parameters, has not yet been written.

Value

`dbetanorm` gives the density, `pbetanorm` gives the distribution function, `qbetanorm` gives the quantile function, and `rbetanorm` generates random deviates.

Author(s)

T. W. Yee

Examples

```
## Not run:
shape1 = 0.1; shape2 = 4; m = 1
x = seq(-10, 2, len=501)
plot(x, dbetanorm(x, shape1, shape2, m=m), type="l", ylim=0:1, las=1,
     ylab=paste("betanorm(",shape1,", ",shape2,", m=",m, ", sd=1)", sep=""),
     main="Blue is density, red is cumulative distribution function",
     sub="Purple lines are the 10,20,...,90 percentiles", col="blue")
lines(x, pbetanorm(x, shape1, shape2, m=m), col="red")
abline(h=0)
probs = seq(0.1, 0.9, by=0.1)
Q = qbetanorm(probs, shape1, shape2, m=m)
lines(Q, dbetanorm(Q, shape1, shape2, m=m), col="purple",
      lty=3, type="h")
lines(Q, pbetanorm(Q, shape1, shape2, m=m), col="purple", lty=3, type="h")
abline(h=probs, col="purple", lty=3)
pbetanorm(Q, shape1, shape2, m=m) - probs # Should be all 0
## End(Not run)
```

`betaprime` *The Beta-Prime Distribution*

Description

Estimation of the two shape parameters of the beta-prime distribution by maximum likelihood estimation.

Usage

```
betaprime(link = "loge", earg=list(), i1 = 2, i2 = NULL, zero = NULL)
```

Arguments

`link`	Parameter link function applied to the two (positive) shape parameters. See `Links` for more choices.
`earg`	List. Extra argument for each of the links. See `earg` in `Links` for general information.
`i1, i2`	Initial values for the first and second shape parameters. A `NULL` value means it is obtained in the `initialize` slot. Note that `i2` is obtained using `i1`.
`zero`	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The value must be from the set {1,2} corresponding respectively to `shape1` and `shape2` respectively. If `zero=NULL` then both parameters are modelled with the explanatory variables.

Details

The beta-prime distribution is given by

$$f(y) = y^{shape1-1}(1+y)^{-shape1-shape2}/B(shape1, shape2)$$

for $y > 0$. The shape parameters are positive, and here, B is the beta function. The mean of Y is $shape1/(shape2-1)$ provided $shape2 > 1$.

If Y has a $Beta(shape1, shape2)$ distribution then $Y/(1-Y)$ and $(1-Y)/Y$ have a $Betaprime(shape1, shape2)$ and $Betaprime(shape2, shape1)$ distribution respectively. Also, if Y_1 has a $gamma(shape1)$ distribution and Y_2 has a $gamma(shape2)$ distribution then Y_1/Y_2 has a $Betaprime(shape1, shape2)$ distribution.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

Note

The response must have positive values only.

The beta-prime distribution is also known as the *beta distribution of the second kind* or the *inverted beta distribution*.

Author(s)

Thomas W. Yee

References

Johnson, N. L. and Kotz, S. and Balakrishnan, N. (1995) Chapter 25 of: *Continuous Univariate Distributions*, 2nd edition, Volume 2, New York: Wiley.

Documentation accompanying the **VGAM** package at `http://www.stat.auckland.ac.nz/~yee` contains further information and examples.

See Also

`betaff`.

Examples

```
yb = rbeta(n <- 1000, shape1=exp(1), shape2=exp(3))
y1 = (1-yb)/yb
y2 = yb/(1-yb)
y3 = rgamma(n, exp(3)) / rgamma(n, exp(2))

fit1 = vglm(y1 ~ 1, betaprime, trace=TRUE)
coef(fit1, matrix=TRUE)

fit2 = vglm(y2 ~ 1, betaprime, trace=TRUE)
coef(fit2, matrix=TRUE)
```

```
fit3 = vglm(y3 ~ 1, betaprime, trace=TRUE)
coef(fit3, matrix=TRUE)

# Compare the fitted values
mean(y3)
fitted(fit3)[1:5]
Coef(fit3)  # Useful for intercept-only models
```

`bilogis4` *Bivariate Logistic Distribution*

Description

Density, distribution function, quantile function and random generation for the 4-parameter bivariate logistic distribution.

Usage

```
dbilogis4(x1, x2, loc1=0, scale1=1, loc2=0, scale2=1)
pbilogis4(q1, q2, loc1=0, scale1=1, loc2=0, scale2=1)
rbilogis4(n, loc1=0, scale1=1, loc2=0, scale2=1)
```

Arguments

`x1, x2, q1, q2`	vector of quantiles.
`n`	number of observations. Must be a positive integer of length 1.
`loc1, loc2`	the location parameters l_1 and l_2.
`scale1, scale2`	the scale parameters s_1 and s_2.

Details

See `bilogis4`, the **VGAM** family function for estimating the four parameters by maximum likelihood estimation, for the formula of the cumulative distribution function and other details.

Value

`dbilogis4` gives the density, `pbilogis4` gives the distribution function, and `rbilogis4` generates random deviates (a two-column matrix).

Author(s)

T. W. Yee

References

Gumbel, E. J. (1961) Bivariate logistic distributions. *Journal of the American Statistical Association*, **56**, 335–349.

See Also

bilogistic4.

Examples

```
## Not run:
par(mfrow=c(1,3))
n = 2000
ymat = rbilogis4(n, loc1=5, loc2=7, scale2=exp(1))
myxlim = c(-2,15)
myylim = c(-10,30)
plot(ymat, xlim=myxlim, ylim=myylim)

N = 100
x1 = seq(myxlim[1], myxlim[2], len=N)
x2 = seq(myylim[1], myylim[2], len=N)
ox = expand.grid(x1, x2)
z = dbilogis4(ox[,1], ox[,2], loc1=5, loc2=7, scale2=exp(1))
contour(x1, x2, matrix(z, N, N), main="density")
z = pbilogis4(ox[,1], ox[,2], loc1=5, loc2=7, scale2=exp(1))
contour(x1, x2, matrix(z, N, N), main="cdf")
## End(Not run)
```

---

bilogistic4 *Bivariate Logistic Distribution Family Function*

Description

Estimates the four parameters of the bivariate logistic distribution by maximum likelihood estimation.

Usage

```
bilogistic4(llocation="identity", lscale="loge",
            iloc1=NULL, iscale1=NULL, iloc2=NULL, iscale2=NULL,
            method.init=1, zero=NULL)
```

Arguments

llocation	Link function applied to both location parameters l_1 and l_2. See Links for more choices.
lscale	Parameter link function applied to both (positive) scale parameters s_1 and s_2. See Links for more choices.

`iloc1, iloc2` Initial values for the location parameters. By default, initial values are chosen internally using `method.init`. Assigning values here will override the argument `method.init`.

`iscale1, iscale2` Initial values for the scale parameters. By default, initial values are chosen internally using `method.init`. Assigning values here will override the argument `method.init`.

`method.init` An integer with value `1` or `2` which specifies the initialization method. If failure to converge occurs try the other value.

`zero` An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The default is none of them. If used, choose values from the set {1,2,3,4}.

Details

The four-parameter bivariate logistic distribution has a density that can be written as

$$f(y_1, y_2; l_1, s_1, l_2, s_2) = 2\frac{\exp[-(y_1 - l_1)/s_1 - (y_2 - l_2)/s_2]}{s_1 s_2 \left(1 + \exp[-(y_1 - l_1)/s_1] + \exp[-(y_2 - l_2)/s_2]\right)^3}$$

where $s_1 > 0$ $s_2 > 0$ are the scale parameters, and l_1 and l_2 are the location parameters. Each of the two responses are unbounded, i.e., $-\infty < y_j < \infty$. The mean of Y_1 is l_1 etc. The fitted values are returned in a 2-column matrix. The cumulative distribution function is

$$F(y_1, y_2; l_1, s_1, l_2, s_2) = \left(1 + \exp[-(y_1 - l_1)/s_1] + \exp[-(y_2 - l_2)/s_2]\right)^{-1}$$

The marginal distribution of Y_1 is

$$P(Y_1 \leq y_1) = F(y_1; l_1, s_1) = \left(1 + \exp[-(y_1 - l_1)/s_1]\right)^{-1}.$$

By default, $\eta_1 = l_1$, $\eta_2 = \log(s_1)$, $\eta_3 = l_2$, $\eta_4 = \log(s_2)$ are the linear/additive predictors.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

Note

This family function uses the BFGS quasi-Newton update formula for the working weight matrices. Consequently the estimated variance-covariance matrix may be inaccurate or simply wrong! The standard errors must be therefore treated with caution; these are computed in functions such as `vcov()` and `summary()`.

Author(s)

T. W. Yee

References

Gumbel, E. J. (1961) Bivariate logistic distributions. *Journal of the American Statistical Association*, **56**, 335–349.

Castillo, E., Hadi, A. S., Balakrishnan, N. Sarabia, J. S. (2005) *Extreme Value and Related Models with Applications in Engineering and Science*, Hoboken, N.J.: Wiley-Interscience.

See Also

logistic, rbilogis4.

Examples

```
ymat = rbilogis4(n <- 1000, loc1=5, loc2=7, scale2=exp(1))
## Not run: plot(ymat)
fit = vglm(ymat ~ 1, fam=bilogistic4, trace=TRUE)
coef(fit, matrix=TRUE)
Coef(fit)
fitted(fit)[1:4,]
vcov(fit)
weights(fit, type="w")[1:4,]
summary(fit)
```

---

binom2.or *Bivariate Logistic Regression*

Description

Fits a Palmgren (bivariate logistic regression) model to two binary responses. Actually, a bivariate logistic/probit/cloglog/cauchit model can be fitted.

Usage

```
binom2.or(lmu = "logit", lmu1 = lmu, lmu2 = lmu, lor = "loge",
          emu=list(), emu1=emu, emu2=emu, eor=list(),
          zero = 3, exchangeable = FALSE, tol = 0.001)
```

Arguments

lmu	Link function applied to the two marginal probabilities. See Links for more choices. See the note below.
lmu1, lmu2	Link function applied to the first and second of the two marginal probabilities.
lor	Link function applied to the odds ratio. See Links for more choices.
emu, emu1, emu2, eor	List. Extra argument for each of the links. See earg in Links for general information.
zero	Which linear/additive predictor is modelled as an intercept only? A NULL means none.

`exchangeable` Logical. If `TRUE`, the two marginal probabilities are constrained to be equal.

`tol` Tolerance for testing independence. Should be some small positive numerical value.

Details

Known also as the *Palmgren model*, the bivariate logistic model is a full-likelihood based model defined as two logistic regressions plus `log(OR) = eta3` where `eta3` is the third linear/additive predictor relating the odds ratio to explanatory variables. Explicitly, the default model is

$$logit[P(Y_j = 1)] = \eta_j, \quad j = 1, 2$$

for the marginals, and

$$\log[P(Y_{00} = 1)P(Y_{11} = 1)/(P(Y_{01} = 1)P(Y_{10} = 1))] = \eta_3,$$

specifies the dependency between the two responses. Here, the responses equal 1 for a success and a 0 for a failure, and the odds ratio is often written $\psi = p_{00}p_{11}/(p_{10}p_{01})$. The model is fitted by maximum likelihood estimation since the full likelihood is specified. The two binary responses are independent if and only if the odds ratio is unity, or equivalently, the log odds ratio is zero.

The default models η_3 as a single parameter only, i.e., an intercept-only model, but this can be circumvented by setting `zero=NULL` to model the odds ratio as a function of all the explanatory variables. The function `binom2.or` can handle `probit`, `cloglog` and `cauchit` links as well, so is quite general. In fact, the two marginal probabilities can each have a different link function. A similar model is the *bivariate probit model* (`binom2.rho`), which is based on a standard bivariate normal distribution, but the bivariate probit model is less interpretable and flexible.

The `exchangeable` argument should be used when the error structure is exchangeable, e.g., with eyes or ears data.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

When fitted, the `fitted.values` slot of the object contains the four joint probabilities, labelled as (Y_1, Y_2) = (0,0), (0,1), (1,0), (1,1), respectively. These estimated probabilities should be extracted with the `fitted` generic function.

Note

The response should be either a 4-column matrix of counts (whose columns correspond to (Y_1, Y_2) = (0,0), (0,1), (1,0), (1,1) respectively), or a two-column matrix where each column has two distinct values.

By default, a constant odds ratio is fitted because `zero=3`. Set `zero=NULL` if you want the odds ratio to be modelled as a function of the explanatory variables; however, numerical problems are more likely to occur.

The argument `lmu`, which is actually redundant, is used for convenience and for upward compatibility: specifying `lmu` only means the link function will be applied to `lmu1` and `lmu2`. Users who want a different link function for each of the two marginal probabilities should use the `lmu1` and `lmu2` arguments, and the argument `lmu` is then ignored. It doesn't make sense to specify `exchangeable=TRUE` and have different link functions for the two marginal probabilities.

Author(s)

Thomas W. Yee

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

le Cessie, S. and van Houwelingen, J. C. (1994) Logistic regression for correlated binary data. *Applied Statistics*, **43**, 95–108.

Palmgren, J. (1989) *Regression Models for Bivariate Binary Responses*. Technical Report no. 101, Department of Biostatistics, University of Washington, Seattle.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

binom2.rho, loglinb2, coalminers, binomialff, logit, probit, cloglog, cauchit.

Examples

```
# Fit the model in Table 6.7 in McCullagh and Nelder (1989)
data(coalminers)
coalminers = transform(coalminers, Age = (age - 42) / 5)
fit = vglm(cbind(nBnW,nBW,BnW,BW) ~ Age, binom2.or(zero=NULL), coalminers)
fitted(fit)
summary(fit)
coef(fit, matrix=TRUE)
## Not run:
attach(coalminers)
matplot(Age, fitted(fit), type="l", las=1, xlab="(age - 42) / 5",
        main=paste("B=Breathlessness, W=Wheeze; 1=(B=0,W=0),",
                   "2=(B=0,W=1), 3=(B=1,W=0), 4=(B=1,W=1)"))
matpoints(Age, fit@y, col=1:4)
detach(coalminers)
## End(Not run)
```

---

binom2.rho *Bivariate Probit Model*

Description

Fits a bivariate probit model to two binary responses.

Usage

```
binom2.rho(lrho = "rhobit", erho=list(),
           init.rho = 0.4, zero = 3, exchangeable = FALSE)
```

Arguments

lrho	Link function applied to the ρ association parameter. See Links for more choices.
erho	List. Extra argument for the lrho link. See earg in Links for general information.
init.rho	Initial value for ρ. This should lie between -1 and 1.
zero	Which linear/additive predictor is modelled as an intercept only? A NULL means none.
exchangeable	Logical. If TRUE, the two marginal probabilities are constrained to be equal.

Details

The *bivariate probit model* was one of the earliest regression models to handle two binary responses jointly. It has a probit link for each of the two marginal probabilities, and models the association between the responses by the ρ parameter of a standard bivariate normal distribution (with zero means and unit variances). One can think of the joint probabilities being $\Phi(\eta_1, \eta_2; \rho)$ where Φ is the cumulative distribution function of a standard bivariate normal distribution with correlation parameter ρ.

The bivariate probit model should not be confused with a *bivariate logit model* with a probit link (see binom2.or). The latter uses the odds ratio to quantify the association. Actually, the bivariate logit model is recommended over the bivariate probit model because the odds ratio is a more natural way of measuring the association between two binary responses.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

When fitted, the fitted.values slot of the object contains the four joint probabilities, labelled as (Y_1, Y_2) = (0,0), (0,1), (1,0), (1,1), respectively.

Note

The response should be either a 4-column matrix of counts (whose columns correspond to (Y_1, Y_2) = (0,0), (0,1), (1,0), (1,1) respectively), or a two-column matrix where each column has two distinct values.

By default, a constant ρ is fitted because zero=3. Set zero=NULL if you want the ρ parameter to be modelled as a function of the explanatory variables. The value ρ lies in the interval $(-1, 1)$, therefore a rhobit link is default.

If converge problems occur, try setting init.rho to some value, e.g., a negative number such as -0.5.

Author(s)

Thomas W. Yee

References

Ashford, J. R. and Sowden, R. R. (1970) Multi-variate probit analysis. *Biometrics*, **26**, 535–546.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

binom2.or, loglinb2, coalminers, binomialff, rhobit, fisherz.

Examples

```
data(coalminers)
coalminers = transform(coalminers, Age = (age - 42) / 5)
fit = vglm(cbind(nBnW,nBW,BnW,BW) ~ Age, binom2.rho, coalminers)
summary(fit)
coef(fit, matrix=TRUE)
```

---

binomialff *Binomial Family Function*

Description

Family function for fitting generalized linear models to binomial responses, where the dispersion parameter may be known or unknown.

Usage

```
binomialff(link = "logit", earg = list(),
           dispersion = 1, mv = FALSE, onedpar = !mv,
           parallel = FALSE, zero = NULL)
```

Arguments

The notation M is used to denote the number of linear/additive predictors.

link	Link function. See Links for more choices.
earg	Extra argument optionally used by the link function. See Links for more information.
dispersion	Dispersion parameter. By default, maximum likelihood is used to estimate the model because it is known. However, the user can specify dispersion = 0 to have it estimated, or else specify a known positive value (or values if mv is TRUE).
mv	Multivariate response? If TRUE, then the response is interpreted as M binary responses, where M is the number of columns of the response matrix. In this case, the response matrix should have zero/one values only. If FALSE and the response is a (2-column) matrix, then the number of successes is given in the first column, and the second column is the number of failures.

onedpar	One dispersion parameter? If mv, then a separate dispersion parameter will be computed for each response (column), by default. Setting onedpar=TRUE will pool them so that there is only one dispersion parameter to be estimated.
parallel	A logical or formula. Used only if mv is TRUE. This argument allows for the parallelism assumption whereby the regression coefficients for a variable is constrained to be equal over the M linear/additive predictors.
zero	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,...,M}, where M is the number of columns of the matrix response.

Details

If the dispersion parameter is unknown, then the resulting estimate is not fully a maximum likelihood estimate (see pp.124–8 of McCullagh and Nelder, 1989).

A dispersion parameter that is less/greater than unity corresponds to under-/over-dispersion relative to the binomial model. Over-dispersion is more common in practice.

Setting mv=TRUE is necessary when fitting a Quadratic RR-VGLM (see cqo) because the response is a matrix of M columns (e.g., one column per species). Then there will be M dispersion parameters (one per column of the response matrix).

When used with cqo and cao, it may be preferable to use the cloglog link.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, vgam, rrvglm, cqo, and cao.

Warning

With a multivariate response, assigning a known dispersion parameter for *each* response is not handled well yet. Currently, only a single known dispersion parameter is handled well.

Note

If mv is FALSE (default), then the response can be of one of three formats: a factor (first level taken as success), a vector of proportions of success, or a 2-column matrix (first column = successes) of counts. The argument weights in the modelling function can also be specified. In particular, for a general vector of proportions, you will need to specify weights because the number of trials is needed.

If mv is TRUE, then the matrix response can only be of one format: a matrix of 1's and 0's (1=success).

The call binomialff(dispersion=0, ...) is equivalent to quasibinomialff(...). The latter was written so that R users of quasibinomial() would only need to add a "ff" to the end of the family function name.

Regardless of whether the dispersion parameter is to be estimated or not, its value can be seen from the output from the summary() of the object.

Author(s)

Thomas W. Yee

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

See Also

quasibinomialff, Links, rrvglm, cqo, cao, zibinomial, dexpbinomial, seq2binomial, binomial.

Examples

```
quasibinomialff()
quasibinomialff(link="probit")

data(hunua)
fit = vgam(agaaus ~ poly(altitude, 2), binomialff(link=cloglog), hunua)
## Not run:
attach(hunua)
plot(altitude, agaaus, col="blue", ylab="P(agaaus=1)",
     main="Presence/absence of Agathis australis", las=1)
o = order(altitude)
lines(altitude[o], fitted(fit)[o], col="red", lwd=2)
detach(hunua)
## End(Not run)
```

---

biplot-methods *Biplot of Constrained Regression Models*

Description

`biplot` is a generic function applied to RR-VGLMs and QRR-VGLMs etc. These apply to rank-1 and rank-2 models of these only. For RR-VGLMs these plot the second latent variable scores against the first latent variable scores.

Methods

x The object from which the latent variables are extracted and/or plotted.

Note

See `lvplot` which is very much related to biplots.

bisa *Birnbaum-Saunders Distribution Family Function*

Description

Estimates the shape and scale parameters of the Birnbaum-Saunders distribution by maximum likelihood estimation.

Usage

```
bisa(lshape = "loge", lscale = "loge",
     eshape = list(), escale = list(),
     ishape = NULL, iscale = 1, method.init = 1, zero = NULL)
```

Arguments

`lscale, lshape`
: Parameter link functions applied to the shape and scale parameters (a and b below). See `Links` for more choices. A log link is the default for both because they are positive.

`escale, eshape`
: List. Extra argument for each of the links. See `earg` in `Links` for general information.

`iscale, ishape`
: Initial values for a and b. A `NULL` means an initial value is chosen internally using `method.init`.

`method.init`
: An integer with value `1` or `2` or `3` which specifies the initialization method. If failure to converge occurs try the other value, or else specify a value for `ishape` and/or `iscale`.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The default is none of them. If used, choose one value from the set {1,2}.

Details

The (two-parameter) Birnbaum-Saunders distribution has a cumulative distribution function that can be written as

$$F(y; a, b) = \Phi[\xi(y/b)/a]$$

where $\Phi(\cdot)$ is the cumulative distribution function of a standard normal (see `pnorm`), $\xi(t) = \sqrt{t} - 1/\sqrt{t}$, $y > 0$, $a > 0$ is the shape parameter, $b > 0$ is the scale parameter. The mean of Y (which is the fitted value) is $b(1 + a^2/2)$. and the variance is $a^2 b^2 (1 + \frac{5}{4}a^2)$. By default, $\eta_1 = \log(a)$ and $\eta_2 = \log(b)$ for this family function.

Note that a and b are orthogonal, i.e., the Fisher information matrix is diagonal. This family function implements Fisher scoring, and it is unnecessary to compute any integrals numerically.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Author(s)

T. W. Yee

References

Lemonte, A. J. and Cribari-Neto, F. and Vasconcellos, K. L. P. (2007) Improved statistical inference for the two-parameter Birnbaum-Saunders distribution. *Computational Statistics & Data Analysis*, **51**, 4656–4681.

Birnbaum, Z. W. and Saunders, S. C. (1969) A new family of life distributions. *Journal of Applied Probability*, **6**, 319–327.

Birnbaum, Z. W. and Saunders, S. C. (1969) Estimation for a family of life distributions with applications to fatigue. *Journal of Applied Probability*, **6**, 328–347.

Engelhardt, M. and Bain, L. J. and Wright, F. T. (1981) Inferences on the parameters of the Birnbaum-Saunders fatigue life distribution based on maximum likelihood estimation. *Technometrics*, **23**, 251–256.

Johnson, N. L. and Kotz, S. and Balakrishnan, N. (1995) *Continuous Univariate Distributions*, 2nd edition, Volume 2, New York: Wiley.

See Also

pbisa, inv.gaussianff.

Examples

```
x = runif(n <- 1000)
y = rbisa(n, shape=exp(-0.5+x), scale=exp(1.5))
fit = vglm(y ~ x, bisa(zero=2), trace=TRUE)
coef(fit, matrix=TRUE)

## Not run:
y = rbisa(n=1000, shape=exp(-0.5), scale=exp(0.5))
fit = vglm(y ~ 1, bisa, trace=TRUE)
hist(y, prob=TRUE, ylim=c(0,0.5), col="lightblue")
coef(fit, matrix=TRUE)
mean(y)
fitted(fit)[1:4]
x = seq(0, max(y), len=200)
lines(x, dbisa(x, Coef(fit)[1], Coef(fit)[2]), col="red", lwd=2)
## End(Not run)
```

Bisa *The Birnbaum-Saunders Distribution*

Description

Density, distribution function, and random generation for the Birnbaum-Saunders distribution.

Usage

```
dbisa(x, shape, scale=1, log=FALSE)
pbisa(q, shape, scale=1)
qbisa(p, shape, scale=1)
rbisa(n, shape, scale=1)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. Must be a positive integer of length 1.
`shape, scale`	the (positive) shape and scale parameters.
`log`	Logical. If `TRUE` then the logarithm of the density is returned.

Details

The Birnbaum-Saunders distribution is a distribution which is used in survival analysis. See `bisa`, the **VGAM** family function for estimating the parameters, for more details.

Value

`dbisa` gives the density, `pbisa` gives the distribution function, and `qbisa` gives the quantile function, and `rbisa` generates random deviates.

Author(s)

T. W. Yee

See Also

`bisa`.

Examples

```
## Not run:
x = seq(0, 6, len=400)
plot(x, dbisa(x, shape=1), type="l", col="blue", ylab="Density", lwd=2,
     main="X ~ Birnbaum-Saunders(shape, scale=1)", ylim=c(0,1.3), lty=3)
lines(x, dbisa(x, shape=2), col="red", lty=2, lwd=2)
lines(x, dbisa(x, shape=0.5), col="green", lty=1, lwd=2)
legend(x=3, y=0.9, legend=paste("shape =",c(0.5,1,2)),
       col=c("green","blue","red"), lty=1:3, lwd=2)

shape = 1
x = seq(0.0, 4, len=401)
plot(x, dbisa(x, shape=shape), type="l", col="blue", las=1, ylab="",
     main="Blue is density, red is cumulative distribution function",
     sub="Purple lines are the 10,20,...,90 percentiles", ylim=0:1)
abline(h=0, col="blue", lty=2)
lines(x, pbisa(x, shape=shape), col="red")
probs = seq(0.1, 0.9, by=0.1)
Q = qbisa(probs, shape=shape)
lines(Q, dbisa(Q, shape=shape), col="purple", lty=3, type="h")
pbisa(Q, shape=shape) - probs    # Should be all zero
abline(h=probs, col="purple", lty=3)
lines(Q, pbisa(Q, shape), col="purple", lty=3, type="h")
## End(Not run)
```

---

bminz *Body Mass Index of New Zealand Adults*

Description

The body mass indexes and ages from an approximate random sample of 700 New Zealand adults.

Usage

```
data(bminz)
```

Format

A data frame with 700 observations on the following 2 variables.

age a numeric vector; their age (years).

BMI a numeric vector; their body mass indexes, which is their weight divided by the square of their height (kg / m^2).

Details

They are a random sample from the Fletcher Challenge/Auckland Heart and Health survey conducted in the early 1990s.

There are some outliers in the data set.

A variable `gender` would be useful, and may be added later.

Source

Clinical Trials Research Unit, University of Auckland, New Zealand.

References

MacMahon, S., Norton, R., Jackson, R., Mackie, M. J., Cheng, A., Vander Hoorn, S., Milne, A., McCulloch, A. (1995) Fletcher Challenge-University of Auckland Heart & Health Study: design and baseline findings. *New Zealand Medical Journal*, **108**, 499–502.

Examples

```
## Not run:
data(bminz)
attach(bminz); plot(age, BMI, col="blue"); detach(bminz)

fit = vgam(BMI ~ s(age, df=c(2,4,2)), fam=lms.yjn, data=bminz, tr=TRUE)
qtplot(fit, pcol="blue", tcol="brown", lcol="brown")
## End(Not run)
```

---

brat *Bradley Terry Model*

Description

Fits a Bradley Terry model (intercept-only model) by maximum likelihood estimation.

Usage

```
brat(refgp = "last", refvalue = 1, init.alpha = 1)
```

Arguments

`refgp`	Integer whose value must be from the set $\{1,\ldots,M+1\}$, where there are $M+1$ competitors. The default value indicates the last competitor is used—but don't input a character string, in general.
`refvalue`	Numeric. A positive value for the reference group.
`init.alpha`	Initial values for the αs. These are recycled to the appropriate length.

Details

The Bradley Terry model involves $M+1$ competitors who either win or lose against each other (no draws/ties allowed in this implementation–see `bratt` if there are ties). The probability that Competitor i beats Competitor j is $\alpha_i/(\alpha_i+\alpha_j)$, where all the αs are positive. Loosely, the αs can be thought of as the competitors' 'abilities'. For identifiability, one of the α_i is set to a known value `refvalue`, e.g., 1. By default, this function chooses the last competitor to have this reference value. The data can be represented in the form of a $M+1$ by $M+1$ matrix of counts, where

winners are the rows and losers are the columns. However, this is not the way the data should be inputted (see below).

Excluding the reference value/group, this function chooses $\log(\alpha_j)$ as the M linear predictors. The log link ensures that the αs are positive.

The Bradley Terry model can be fitted by logistic regression, but this approach is not taken here. The Bradley Terry model can be fitted with covariates, e.g., a home advantage variable, but unfortunately, this lies outside the VGLM theoretical framework and therefore cannot be handled with this code.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`.

Warning

Presently, the residuals are wrong, and the prior weights are not handled correctly. Ideally, the total number of counts should be the prior weights, after the response has been converted to proportions. This would make it similar to family functions such as `multinomial` and `binomialff`.

Note

The function `Brat` is useful for coercing a $M+1$ by $M+1$ matrix of counts into a one-row matrix suitable for `brat`. Diagonal elements are skipped, and the usual S order of `c(a.matrix)` of elements is used. There should be no missing values apart from the diagonal elements of the square matrix. The matrix should have winners as the rows, and losers as the columns. In general, the response should be a 1-row matrix with $M(M+1)$ columns.

Only an intercept model is recommended with `brat`. It doesn't make sense really to include covariates because of the limited VGLM framework.

Notationally, note that the **VGAM** family function `brat` has $M+1$ contestants, while `bratt` has M contestants.

Author(s)

T. W. Yee

References

Agresti, A. (2002) *Categorical Data Analysis*, 2nd ed. New York: Wiley.

Stigler, S. (1994) Citation patterns in the journals of statistics and probability. *Statistical Science*, **9**, 94–108.

The `BradleyTerry` package has more comprehensive capabilities than this function.

See Also

`bratt`, `Brat`, `multinomial`, `binomialff`.

Examples

```
# citation statistics: being cited is a 'win'; citing is a 'loss'
journal = c("Biometrika", "Comm Statist", "JASA", "JRSS-B")
m = matrix(c( NA,  33, 320, 284,
              730, NA, 813, 276,
              498, 68,  NA, 325,
              221, 17, 142,  NA), 4,4)
dimnames(m) = list(winner = journal, loser = journal)
fit = vglm(Brat(m) ~ 1, brat(refgp=1), trace=TRUE)
fit = vglm(Brat(m) ~ 1, brat(refgp=1), trace=TRUE, cri="c")
summary(fit)
c(0, coef(fit)) # log-abilities (in order of "journal")
c(1, Coef(fit)) # abilities (in order of "journal")
fitted(fit)  # probabilities of winning in awkward form
(check = InverseBrat(fitted(fit)))  # probabilities of winning
check + t(check)   # Should be 1's in the off-diagonals
```

---

bratt *Bradley Terry Model With Ties*

Description

Fits a Bradley Terry model with ties (intercept-only model) by maximum likelihood estimation.

Usage

```
bratt(refgp = "last", refvalue = 1, init.alpha = 1, i0 = 0.01)
```

Arguments

`refgp`	Integer whose value must be from the set {1,...,M}, where there are M competitors. The default value indicates the last competitor is used—but don't input a character string, in general.
`refvalue`	Numeric. A positive value for the reference group.
`init.alpha`	Initial values for the αs. These are recycled to the appropriate length.
`i0`	Initial value for α_0. If convergence fails, try another positive value.

Details

There are several models that extend the ordinary Bradley Terry model to handle ties. This family function implements one of these models. It involves M competitors who either win or lose or tie against each other. (If there are no draws/ties then use `brat`). The probability that Competitor i beats Competitor j is $\alpha_i/(\alpha_i + \alpha_j + \alpha_0)$, where all the αs are positive. The probability that Competitor i ties with Competitor j is $\alpha_0/(\alpha_i + \alpha_j + \alpha_0)$. Loosely, the αs can be thought of as the competitors' 'abilities', and α_0 is an added parameter to model ties. For identifiability, one of the α_i is set to a known value `refvalue`, e.g., 1. By default, this function chooses the last competitor to have this reference value. The data can be represented in the form of a M by M matrix of counts,

where winners are the rows and losers are the columns. However, this is not the way the data should be inputted (see below).

Excluding the reference value/group, this function chooses $\log(\alpha_j)$ as the first $M-1$ linear predictors. The log link ensures that the αs are positive. The last linear predictor is $\log(\alpha_0)$.

The Bradley Terry model can be fitted with covariates, e.g., a home advantage variable, but unfortunately, this lies outside the VGLM theoretical framework and therefore cannot be handled with this code.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`.

Note

The function `Brat` is useful for coercing a M by M matrix of counts into a one-row matrix suitable for `bratt`. Diagonal elements are skipped, and the usual S order of `c(a.matrix)` of elements is used. There should be no missing values apart from the diagonal elements of the square matrix. The matrix should have winners as the rows, and losers as the columns. In general, the response should be a matrix with $M(M-1)$ columns.

Also, a symmetric matrix of ties should be passed into `Brat`. The diagonal of this matrix should be all `NA`s.

Only an intercept model is recommended with `bratt`. It doesn't make sense really to include covariates because of the limited VGLM framework.

Notationally, note that the **VGAM** family function `brat` has $M+1$ contestants, while `bratt` has M contestants.

Author(s)

T. W. Yee

References

Torsney, B. (2004) Fitting Bradley Terry models using a multiplicative algorithm. In: Antoch, J. (ed.) *Proceedings in Computational Statistics COMPSTAT 2004*, Physica-Verlag: Heidelberg. Pages 513–526.

See Also

`brat`, `Brat`, `binomialff`.

Examples

```
# citation statistics: being cited is a 'win'; citing is a 'loss'
journal = c("Biometrika", "Comm Statist", "JASA", "JRSS-B")
m = matrix(c( NA, 33, 320, 284,
              730, NA, 813, 276,
              498, 68,  NA, 325,
              221, 17, 142,  NA), 4,4)
```

```
dimnames(m) = list(winner = journal, loser = journal)

# Add some ties. This is fictitional data.
ties = 5 + 0*m
ties[2,1] = ties[1,2] = 9

# Now fit the model
fit = vglm(Brat(m, ties) ~ 1, bratt(refgp=1), trace=TRUE)
fit = vglm(Brat(m, ties) ~ 1, bratt(refgp=1), trace=TRUE, cri="c")

summary(fit)
c(0, coef(fit)) # log-abilities (in order of "journal"); last is log(alpha0)
c(1, Coef(fit)) # abilities (in order of "journal"); last is alpha0

fit@misc$alpha  # alpha_1,...,alpha_M
fit@misc$alpha0 # alpha_0

fitted(fit)  # probabilities of winning and tying, in awkward form
predict(fit)
(check = InverseBrat(fitted(fit)))    # probabilities of winning
qprob = attr(fitted(fit), "probtie")  # probabilities of a tie
qprobmat = InverseBrat(c(qprob), NCo=nrow(ties))  # probabilities of a tie
check + t(check) + qprobmat    # Should be 1's in the off-diagonals
```

---

calibrate-methods *Calibration for Constrained Regression Models*

Description

`calibrate` is a generic function applied to QRR-VGLMs and RR-VGAMs etc.

Methods

object The object from which the calibration is performed.

---

calibrate *Model Calibrations*

Description

`calibrate` is a generic function used to produce calibrations from various model fitting functions. The function invokes particular 'methods' which depend on the 'class' of the first argument.

Usage

```
calibrate(object, ...)
```

Arguments

object	An object for which a calibration is desired.
...	Additional arguments affecting the calibration produced. Usually the most important argument in ... is newdata which, for calibrate, contains new *response* data, **Y**, say.

Details

Given a regression model with explanatory variables **X** and response **Y**, calibration involves estimating **X** from **Y** using the regression model. It can be loosely thought of as the opposite of predict (which takes an **X** and returns a **Y**.)

Value

In general, given a new response **Y**, the explanatory variables **X** are returned. However, for constrained ordination models such as CQO and CAO models, it is usually not possible to return **X**, so the latent variables are returned instead (they are linear combinations of the **X**). See the specific calibrate methods functions to see what they return.

Note

This function was not called predictx because of the inability of constrained ordination models to return **X**; they can only return the latent variable values (site scores) instead.

Author(s)

T. W. Yee

See Also

predict, calibrate.qrrvglm.

Examples

```
data(hspider)
hspider[,1:6] = scale(hspider[,1:6]) # Standardized environmental vars
set.seed(123)
p1 = cao(cbind(Pardlugu, Pardmont, Pardnigr, Pardpull, Zoraspin) ~
         WaterCon + BareSand + FallTwig +
         CoveMoss + CoveHerb + ReflLux,
         family = poissonff, data = hspider, Rank = 1,
         df1.nl = c(Zoraspin=2, 1.9),
         Bestof = 3, Crow1positive = TRUE)

siteNos = 1:2  # Calibrate these sites
cp1 = calibrate(p1, new=data.frame(p1@y[siteNos,]), trace=TRUE)

## Not run:
# Graphically compare the actual site scores with their calibrated values
persp(p1, main="Solid=actual, dashed=calibrated site scores",
```

```
      label=TRUE, col="blue", las=1)
abline(v=lv(p1)[siteNos], lty=1, col=1:length(siteNos)) # actual site scores
abline(v=cp1, lty=2, col=1:length(siteNos)) # calibrated values
## End(Not run)
```

`calibrate.qrrvglm` *Calibration for CQO, UQO and CAO models*

Description

Performs maximum likelihood calibration for constrained and unconstrained quadratic and additive ordination models (CQO and CAO models are better known as QRR-VGLMs and RR-VGAMs respectively).

Usage

```
calibrate.qrrvglm(object, newdata = NULL,
        type=c("lv","predictors","response","vcov","all3or4"),
        initial.vals = NULL, ...)
```

Arguments

`object`	The fitted CQO/CAO model.
`newdata`	A data frame with new response data (usually new species data). The default is to use the original data used to fit the model; however, the calibration may take a long time to compute because the computations are expensive.
`type`	What type of result is to be returned. The first are the calibrated latent variables or site scores. This must be computed always. The `"predictors"` are the linear/quadratic or additive predictors evaluated at the calibrated latent variables or site scores. The `"response"` are the fitted means evaluated at the calibrated latent variables or site scores. The `"vcov"` are the estimated variance-covariance matrices of the calibrated latent variables or site scores. The `"all3or4"` is for all of them, i.e., all `types`. For CAO models, `"vcov"` is unavailable, so all 3 are returned. For CQO models, `"vcov"` is available, so all 4 are returned.
`initial.vals`	Initial values for the search. For rank-1 models, this should be a vector of length `nrow(newdata)`, and for rank 2 models this should be a two column matrix with the number of rows equalling the number of rows in `newdata`. The default is a grid defined by arguments in `calibrate.qrrvglm.control`.
`...`	Arguments that are fed into `calibrate.qrrvglm.control`.

Details

Given a fitted regression CQO/CAO model, maximum likelihood calibration is theoretically easy and elegant. However, the method assumes that all species are independent, which is not really true in practice. More details and references are given in Yee (2005).

The function `optim` is used to search for the maximum likelihood solution. Good initial values are needed, and `calibrate.qrrvglm.control` allows the user some control over the choice of these.

Value

The argument `type` determines what is returned. If `type="all3or4"` then all the `type` values are returned in a list, with the following components. Each component has length `nrow(newdata)`.

`lv`	Calibrated latent variables or site scores.
`predictors`	linear/quadratic or additive predictors. For example, for Poisson families, this will be on a log scale, and for binomial families, this will be on a logit scale.
`response`	Fitted values of the response, evaluated at the calibrated latent variables or site scores.
`vcov`	Estimated variance-covariance matrix of the calibrated latent variables or site scores. Actually, these are stored in an array whose last dimension is `nrow(newdata)`.

Warning

This function is computationally expensive. Setting `trace=TRUE` to get a running log is a good idea.

Note

Despite the name of this function, UQO and CAO models are handled as well.

Author(s)

T. W. Yee

References

Yee, T. W. (2005) On constrained and unconstrained quadratic ordination. *Manuscript in preparation*.

ter Braak, C. J. F. 1995. Calibration. In: *Data Analysis in Community and Landscape Ecology* by Jongman, R. H. G., ter Braak, C. J. F. and van Tongeren, O. F. R. (Eds.) Cambridge University Press, Cambridge.

See Also

`calibrate.qrrvglm.control`, `calibrate`, `cqo`, `uqo`, `cao`.

Examples

```
data(hspider)
hspider[,1:6] = scale(hspider[,1:6]) # Standardize the environmental variables
set.seed(123)
p1 = cqo(cbind(Pardlugu, Pardmont, Pardnigr, Pardpull, Zoraspin) ~
         WaterCon + BareSand + FallTwig +
         CoveMoss + CoveHerb + ReflLux,
         family = poissonff, data = hspider, Rank = 1,
         ITol = TRUE, Crow1positive = TRUE)

siteNos = 1:2  # Calibrate these sites
cp1 = calibrate(p1, new=data.frame(p1@y[siteNos,]), trace=TRUE)

## Not run:
# Graphically compare the actual site scores with their calibrated values
persp(p1, main="Site scores: solid=actual, dashed=calibrated",
      label=TRUE, col="blue", las=1)
abline(v=lv(p1)[siteNos], lty=1, col=1:length(siteNos)) # actual site scores
abline(v=cp1, lty=2, col=1:length(siteNos)) # calibrated values
## End(Not run)
```

---

calibrate.qrrvglm.control

Control function for CQO/UQO/CAO calibration

Description

Algorithmic constants and parameters for running `calibrate.qrrvglm` are set using this function.

Usage

```
calibrate.qrrvglm.control(object, trace = FALSE, Method.optim = "BFGS",
                          gridSize = if (Rank == 1) 9 else 5,
                          varlvI = FALSE, ...)
```

Arguments

`object`	The fitted CQO/UQO/CAO model. The user should ignore this argument.
`trace`	Logical indicating if output should be produced for each iteration. It is a good idea to set this argument to be `TRUE` since the computations are expensive.
`Method.optim`	Character. Fed into the `method` argument of `optim`.
`gridSize`	Numeric, recycled to length `Rank`. Controls the resolution of the grid used for initial values. For each latent variable, an equally spaced grid of length `gridSize` is cast from the smallest site score to the largest site score. Then the likelihood function is evaluated on the grid, and the best fit is chosen as the initial value. Thus increasing the value of `gridSize` increases the chance of obtaining the global solution, however, the computing time increases proportionately.

varlvI	Logical. For CQO objects only, this argument is fed into Coef.qrrvglm.
...	Avoids an error message for extraneous arguments.

Details

Most CQO/CAO users will only need to make use of trace and gridSize. These arguments should be used inside their call to calibrate.qrrvglm, not this function directly.

Value

A list which with the following components.

trace	Numeric (even though the input can be logical).
gridSize	Positive integer.
varlvI	Logical.

Note

Despite the name of this function, UQO and CAO models are handled as well.

Author(s)

T. W. Yee

References

Yee, T. W. (2005) On constrained and unconstrained quadratic ordination. *Manuscript in preparation*.

See Also

calibrate.qrrvglm, Coef.qrrvglm.

Examples

```
data(hspider)
hspider[,1:6] = scale(hspider[,1:6]) # Needed when ITol = TRUE
set.seed(123)
p1 = cqo(cbind(Alopacce, Alopcune, Pardlugu, Pardnigr,
               Pardpull, Trocterr, Zoraspin) ~
         WaterCon + BareSand + FallTwig +
         CoveMoss + CoveHerb + ReflLux,
         family = poissonff, data = hspider, ITol = TRUE)
sort(p1@misc$deviance.Bestof) # A history of all the iterations

siteNos = 1:2  # Calibrate these sites
cp1 = calibrate(p1, new=data.frame(p1@y[siteNos,]), trace=TRUE)

## Not run:
# Graphically compare the actual site scores with their calibrated values
persp(p1, main="Site scores: solid=actual, dashed=calibrated",
```

```
      label=TRUE, col="blue", las=1)
abline(v=lv(p1)[siteNos], lty=1, col=1:length(siteNos)) # actual site scores
abline(v=cp1, lty=2, col=1:length(siteNos)) # calibrated values
## End(Not run)
```

---

cao *Fitting Constrained Additive Ordination (CAO)*

Description

A constrained additive ordination (CAO) model is fitted using the *reduced-rank vector generalized additive model* (RR-VGAM) framework.

Usage

```
cao(formula, family, data = list(),
    weights = NULL, subset = NULL, na.action = na.fail,
    etastart = NULL, mustart = NULL, coefstart = NULL,
    control = cao.control(...), offset = NULL,
    method = "cao.fit", model = FALSE, x.arg = TRUE, y.arg = TRUE,
    contrasts = NULL, constraints = NULL,
    extra = NULL, qr.arg = FALSE, smart = TRUE, ...)
```

Arguments

The arguments of cao are a mixture of those from vgam and cqo, but with some extras in cao.control. Currently, not all of the following arguments work properly.

formula	a symbolic description of the model to be fit. The RHS of the formula is used to construct the latent variables, upon which the smooths are applied. All the variables in the formula are used for the construction of latent variables except for those specified by the argument Norrr, which is itself a formula. The LHS of the formula contains the response variables, which should be a matrix with each column being a response (species).
family	a function of class "vglmff" describing what statistical model is to be fitted. See cqo for a list of those presently implemented.
data	an optional data frame containing the variables in the model. By default the variables are taken from environment(formula), typically the environment from which cao is called.
weights	an optional vector or matrix of (prior) weights to be used in the fitting process. For cao, this argument currently should not be used.
subset	an optional logical vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The "factory-fresh" default is na.omit.

etastart	starting values for the linear predictors. It is a M-column matrix. If $M = 1$ then it may be a vector. For cao, this argument currently should not be used.
mustart	starting values for the fitted values. It can be a vector or a matrix. Some family functions do not make use of this argument. For cao, this argument currently should not be used.
coefstart	starting values for the coefficient vector. For cao, this argument currently should not be used.
control	a list of parameters for controlling the fitting process. See cao.control for details.
offset	a vector or M-column matrix of offset values. These are *a priori* known and are added to the linear predictors during fitting. For cao, this argument currently should not be used.
method	the method to be used in fitting the model. The default (and presently only) method cao.fit uses iteratively reweighted least squares (IRLS) within FORTRAN code called from optim.
model	a logical value indicating whether the *model frame* should be assigned in the model slot.
x.arg, y.arg	logical values indicating whether the model matrix and response vector/matrix used in the fitting process should be assigned in the x and y slots. Note the model matrix is the linear model (LM) matrix.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
constraints	an optional list of constraint matrices. For cao, this argument currently should not be used. The components of the list must be named with the term it corresponds to (and it must match in character format). Each constraint matrix must have M rows, and be of full-column rank. By default, constraint matrices are the M by M identity matrix unless arguments in the family function itself override these values. If constraints is used it must contain *all* the terms; an incomplete list is not accepted.
extra	an optional list with any extra information that might be needed by the family function. For cao, this argument currently should not be used.
qr.arg	For cao, this argument currently should not be used.
smart	logical value indicating whether smart prediction (smartpred) will be used.
...	further arguments passed into cao.control.

Details

CAO can be loosely be thought of as the result of fitting generalized additive models (GAMs) to several responses (e.g., species) against a very small number of latent variables. Each latent variable is a linear combination of the explanatory variables; the coefficients $\mathbf{C}$ (called C below) are called *constrained coefficients* or *canonical coefficients*, and are interpreted as weights or loadings. The $\mathbf{C}$ are estimated by maximum likelihood estimation. It is often a good idea to apply scale to each explanatory variable first.

For each response (e.g., species), each latent variable is smoothed by a cubic smoothing spline, thus CAO is data-driven. If each smooth were a quadratic then CAO would simplify to *constrained quadratic ordination* (CQO; formerly called *canonical Gaussian ordination* or CGO). If

each smooth were linear then CAO would simplify to *constrained linear ordination* (CLO). CLO can theoretically be fitted with `cao` by specifying `df1.nl=0`, however it is more efficient to use `rrvglm`.

Currently, only `Rank=1` is implemented, and only `Norrr = ~1` models are handled.

With binomial data, the default formula is

$$logit(P[Y_s = 1]) = \eta_s = f_s(\nu), \quad s = 1, 2, \ldots, S$$

where x_2 is a vector of environmental variables, and $\nu = C^T x_2$ is a R-vector of latent variables. The η_s is an additive predictor for species s, and it models the probabilities of presence as an additive model on the logit scale. The matrix C is estimated from the data, as well as the smooth functions f_s. The argument `Norrr = ~ 1` specifies that the vector x_1, defined for RR-VGLMs and QRR-VGLMs, is simply a 1 for an intercept. Here, the intercept in the model is absorbed into the functions. A `cloglog` link may be preferable over a `logit` link.

With Poisson count data, the formula is

$$\log(E[Y_s]) = \eta_s = f_s(\nu)$$

which models the mean response as an additive models on the log scale.

The fitted latent variables (site scores) are scaled to have unit variance. The concept of a tolerance is undefined for CAO models, but the optima and maxima are defined. The generic functions `Max` and `Opt` should work for CAO objects, but note that if the maximum occurs at the boundary then `Max` will return a `NA`. Inference for CAO models is currently undeveloped.

Value

An object of class `"cao"` (this may change to `"rrvgam"` in the future). Several generic functions can be applied to the object, e.g., `Coef`, `ccoef`, `lvplot`, `summary`.

Warning

CAO models present a difficult optimization problem, partly because the log-likelihood function contains many local solutions. To obtain the (global) solution the user is advised to try many initial values. This can be done by setting `Bestof` some appropriate value (see `cao.control`). Trying many initial values becomes progressively more important as the nonlinear degrees of freedom of the smooths increase. Use `set.seed` to make your results reproducible.

Currently the dispersion parameter for a `gaussianff` CAO model is estimated slightly differently and may be slightly biassed downwards (usually a little too small).

Note

CAO models are computationally expensive, therefore setting `trace = TRUE` is a good idea, as well as running it on a simple random sample of the data set instead.

Sometimes the IRLS algorithm does not converge within the FORTRAN code. This results in warnings being issued. In particular, if an error code of 3 is issued, then this indicates the IRLS algorithm has not converged. One possible remedy is to increase or decrease the nonlinear degrees of freedom so that the curves become more or less flexible, respectively.

Author(s)

T. W. Yee

References

Yee, T. W. (2006) Constrained additive ordination. *Ecology*, **87**, 203–213.

See Also

cao.control, Coef.cao, cqo, lv, Opt, Max, lv, persp.cao, poissonff, binomialff, negbinomial, gamma2, gaussianff, set.seed, gam.

Examples

```
data(hspider)
hspider[,1:6] = scale(hspider[,1:6]) # Standardized environmental vars
set.seed(149)
ap1 = cao(cbind(Pardlugu, Pardmont, Pardnigr, Pardpull) ~
          WaterCon + BareSand + FallTwig +
          CoveMoss + CoveHerb + ReflLux,
          family = poissonff, data = hspider, Rank = 1,
          df1.nl = c(Pardpull=2.7, 2.5),
          Bestof = 7, Crow1positive = FALSE)
sort(ap1@misc$deviance.Bestof) # A history of all the iterations

Coef(ap1)
ccoef(ap1)

## Not run:
par(mfrow=c(2,2))
plot(ap1)   # All the curves are unimodal; some quite symmetric

par(mfrow=c(1,1), las=1)
index = 1:ncol(ap1@y)
lvplot(ap1, lcol=index, pcol=index, y=TRUE)

trplot(ap1, label=TRUE, col=index)
abline(a=0, b=1, lty=2)

trplot(ap1, label=TRUE, col="blue", log="xy", whichSp=c(1,3))
abline(a=0, b=1, lty=2)

persp(ap1, col=index, lwd=2, label=TRUE)
abline(v=Opt(ap1), lty=2, col=index)
abline(h=Max(ap1), lty=2, col=index)
## End(Not run)
```

`cao.control` *Control Function for RR-VGAMs (CAO)*

Description

Algorithmic constants and parameters for a constrained additive ordination (CAO), by fitting a *reduced-rank vector generalized additive model* (RR-VGAM), are set using this function. This is the control function for `cao`.

Usage

```
cao.control(Rank=1, all.knots = FALSE,
            criterion="deviance",
            Cinit=NULL,
            Crow1positive=TRUE,
            epsilon = 1.0e-05,
            Etamat.colmax = 10,
            GradientFunction=FALSE,
            iKvector = 0.1,
            iShape = 0.1,
            Norrr = ~ 1,
            SmallNo = 5.0e-13,
            Use.Init.Poisson.QO=TRUE,
            Bestof = if(length(Cinit)) 1 else 10,
            maxitl = 40,
            method.init = 1,
            bf.epsilon = 1.0e-7,
            bf.maxit = 40,
            Maxit.optim = 250,
            optim.maxit = 20,
            SD.sitescores = 1.0,
            SD.Cinit = 0.02,
            trace = TRUE,
            df1.nl = 2.5, df2.nl = 2.5,
            spar1 = 0, spar2 = 0, ...)
```

Arguments

Many of these arguments are identical to `qrrvglm.control`. Here, R is the `Rank`, M is the number of additive predictors, and S is the number of responses (species). Thus $M = S$ for binomial and Poisson responses, and $M = 2S$ for the negative binomial and 2-parameter gamma distributions.

`Rank`	The numerical rank R of the model, i.e., the number of latent variables. Currently only `Rank=1` is implemented.
`all.knots`	Logical indicating if all distinct points of the smoothing variables are to be used as knots. Assigning the value `FALSE` means fewer knots are chosen when the

number of distinct points is large, meaning less computational expense. See vgam.control for details.

criterion — Convergence criterion. Currently, only one is supported: the deviance is minimized.

Cinit — Optional initial **C** matrix which may speed up convergence.

Crow1positive — Logical vector of length Rank (recycled if necessary): are the elements of the first row of **C** positive? For example, if Rank is 4, then specifying Crow1positive=c(FALSE, TRUE) will force **C[1,1]** and **C[1,3]** to be negative, and **C[1,2]** and **C[1,4]** to be positive.

epsilon — Positive numeric. Used to test for convergence for GLMs fitted in FORTRAN. Larger values mean a loosening of the convergence criterion.

Etamat.colmax — Positive integer, no smaller than Rank. Controls the amount of memory used by .Init.Poisson.QO(). It is the maximum number of columns allowed for the pseudo-response and its weights. In general, the larger the value, the better the initial value. Used only if Use.Init.Poisson.QO=TRUE.

GradientFunction — Logical. Whether optim's argument gr is used or not, i.e., to compute gradient values. Used only if FastAlgorithm is TRUE. Currently, this argument must be set to FALSE.

iKvector, iShape — See qrrvglm.control.

Norrr — Formula giving terms that are *not* to be included in the reduced-rank regression (or formation of the latent variables). The default is to omit the intercept term from the latent variables. Currently, only Norrr = ~ 1 is implemented.

SmallNo — Positive numeric between .Machine$double.eps and 0.0001. Used to avoid under- or over-flow in the IRLS algorithm.

Use.Init.Poisson.QO — Logical. If TRUE then the function .Init.Poisson.QO is used to obtain initial values for the canonical coefficients **C**. If FALSE then random numbers are used instead.

Bestof — Integer. The best of Bestof models fitted is returned. This argument helps guard against local solutions by (hopefully) finding the global solution from many fits. The argument works only when the function generates its own initial value for **C**, i.e., when **C** are *not* passed in as initial values. The default is only a convenient minimal number and users are urged to increase this value.

maxitl — Positive integer. Maximum number of Newton-Raphson/Fisher-scoring/local-scoring iterations allowed.

method.init — See qrrvglm.control.

bf.epsilon — Positive numeric. Tolerance used by the modified vector backfitting algorithm for testing convergence.

bf.maxit — Positive integer. Number of backfitting iterations allowed in the compiled code.

Maxit.optim — Positive integer. Number of iterations given to the function optim at each of the optim.maxit iterations.

`optim.maxit` Positive integer. Number of times `optim` is invoked.

`SD.sitescores`
Numeric. Standard deviation of the initial values of the site scores, which are generated from a normal distribution. Used when `Use.Init.Poisson.QO` is `FALSE`.

`SD.Cinit` Standard deviation of the initial values for the elements of **C**. These are normally distributed with mean zero. This argument is used only if `Use.Init.Poisson.QO = FALSE`.

`trace` Logical indicating if output should be produced for each iteration. Having the value `TRUE` is a good idea for large data sets.

`df1.nl, df2.nl`
Numeric and non-negative, recycled to length *S*. Nonlinear degrees of freedom for smooths of the first and second latent variables. A value of 0 means the smooth is linear. Roughly, a value between 1.0 and 2.0 often has the approximate flexibility of a quadratic. The user should not assign too large a value to this argument, e.g., the value 4.0 is probably too high. The argument `df1.nl` is ignored if `spar1` is assigned a positive value or values. Ditto for `df2.nl`.

`spar1, spar2` Numeric and non-negative, recycled to length *S*. Smoothing parameters of the smooths of the first and second latent variables. The larger the value, the more smooth (less wiggly) the fitted curves. These arguments are an alternative to specifying `df1.nl` and `df2.nl`. A value 0 (the default) for `spar1` means that `df1.nl` is used. Ditto for `spar2`. The values are on a scaled version of the latent variables. See Green and Silverman (1994) for more information.

`...` Ignored at present.

Details

Allowing the smooths too much flexibility means the CAO optimization problem becomes more difficult to solve. This is because the number of local solutions increases as the nonlinearity of the smooths increases. In situations of high nonlinearity, many initial values should be used, so that `Bestof` should be assigned a larger value. In general, there should be a reasonable value of `df1.nl` somewhere between 0 and about 3 for most data sets.

Value

A list with the components corresponding to its arguments, after some basic error checking.

Note

The argument `df1.nl` can be inputted in the format `c(spp1=2, spp2=3, 2.5)`, say, meaning the default value is 2.5, but two species have alternative values.

If `spar1=0` and `df1.nl=0` then this represents fitting linear functions (CLO). Currently, this is handled in the awkward manner of setting `df1.nl` to be a small positive value, so that the smooth is almost linear but not quite. A proper fix to this special case should done in the short future.

Author(s)

T. W. Yee

References

Yee, T. W. (2006) Constrained additive ordination. *Ecology*, **87**, 203–213.

Green, P. J. and Silverman, B. W. (1994) *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach*, London: Chapman & Hall.

See Also

cao.

Examples

```
## Not run:
data(hspider)
hspider[,1:6] = scale(hspider[,1:6]) # Standardized environmental vars
set.seed(123)
ap1 = cao(cbind(Pardlugu, Pardmont, Pardnigr, Pardpull, Zoraspin) ~
          WaterCon + BareSand + FallTwig +
          CoveMoss + CoveHerb + ReflLux,
          family = poissonff, data = hspider,
          df1.nl = c(Zoraspin=2.3, 2.1),
          Bestof = 10, Crow1positive = FALSE)
sort(ap1@misc$deviance.Bestof) # A history of all the iterations

Coef(ap1)

par(mfrow=c(2,3)) # All or most of the curves are unimodal; some are
plot(ap1, lcol="blue") # quite symmetric. Hence a CQO model should be ok

par(mfrow=c(1,1), las=1)
index = 1:ncol(ap1@y)  # lvplot is jagged because only 28 sites
lvplot(ap1, lcol=index, pcol=index, y=TRUE)

trplot(ap1, label=TRUE, col=index)
abline(a=0, b=1, lty=2)

persp(ap1, label=TRUE, col=1:4)
## End(Not run)
```

---

cauchit *Cauchit Link Function*

Description

Computes the cauchit (tangent) link transformation, including its inverse and the first two derivatives.

Usage

```
cauchit(theta, earg = list(bvalue= .Machine$double.eps),
        inverse = FALSE, deriv = 0, short = TRUE, tag = FALSE)
```

Arguments

theta	Numeric or character. See below for further details.
earg	List. Extra argument for passing in additional information. Values of theta which are less than or equal to 0 can be replaced by the bvalue component of the list earg before computing the link function value. Values of theta which are greater than or equal to 1 can be replaced by 1 minus the bvalue component of the list earg before computing the link function value. The component name bvalue stands for "boundary value". See Links for general information about earg.
inverse	Logical. If TRUE the inverse function is computed.
deriv	Order of the derivative. Integer with value 0, 1 or 2.
short	Used for labelling the blurb slot of a vglmff-class object.
tag	Used for labelling the linear/additive predictor in the initialize slot of a vglmff-class object. Contains a little more information if TRUE.

Details

This link function is an alternative link function for parameters that lie in the unit interval. This type of link bears the same relation to the Cauchy distribution as the probit link bears to the Gaussian. One characteristic of this link function is that the tail is heavier relative to the other links (see examples below).

Numerical values of theta close to 0 or 1 or out of range result in Inf, -Inf, NA or NaN. The arguments short and tag are used only if theta is character.

Value

For deriv = 0, the tangent of theta, i.e., tan(pi * (theta-0.5)) when inverse = FALSE, and if inverse = TRUE then 0.5 + atan(theta)/pi.

For deriv = 1, then the function returns d theta / d eta as a function of theta if inverse = FALSE, else if inverse = TRUE then it returns the reciprocal.

Note

Numerical instability may occur when theta is close to 1 or 0. One way of overcoming this is to use earg.

As mentioned above, in terms of the threshold approach with cumulative probabilities for an ordinal response this link function corresponds to the Cauchy distribution (see cauchy1).

Author(s)

Thomas W. Yee

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

See Also

logit, probit, cloglog, loge, cauchy1.

Examples

```
p = seq(0.01, 0.99, by=0.01)
cauchit(p)
max(abs(cauchit(cauchit(p), inverse=TRUE) - p)) # Should be 0

p = c(seq(-0.02, 0.02, by=0.01), seq(0.97, 1.02, by=0.01))
cauchit(p)  # Has no NAs

## Not run:
par(mfrow=c(2,2))
y = seq(-4, 4, length=100)

for(d in 0:1) {
    matplot(p, cbind(logit(p, deriv=d), probit(p, deriv=d)),
            type="n", col="purple", ylab="transformation",
            lwd=2, las=1, main=if(d==0) "Some probability link functions"
            else "First derivative")
    lines(p, logit(p, deriv=d), col="limegreen", lwd=2)
    lines(p, probit(p, deriv=d), col="purple", lwd=2)
    lines(p, cloglog(p, deriv=d), col="chocolate", lwd=2)
    lines(p, cauchit(p, deriv=d), col="tan", lwd=2)
    if(d==0) {
        abline(v=0.5, h=0, lty="dashed")
        legend(0, 4.5, c("logit", "probit", "cloglog", "cauchit"),
               col=c("limegreen","purple","chocolate", "tan"), lwd=2)
    } else
        abline(v=0.5, lty="dashed")
}

for(d in 0) {
    matplot(y, cbind(logit(y, deriv=d, inverse=TRUE),
                     probit(y, deriv=d, inverse=TRUE)),
            type="n", col="purple", xlab="transformation", ylab="p",
            main=if(d==0) "Some inverse probability link functions"
            else "First derivative", lwd=2, las=1)
    lines(y, logit(y, deriv=d, inverse=TRUE), col="limegreen", lwd=2)
    lines(y, probit(y, deriv=d, inverse=TRUE), col="purple", lwd=2)
    lines(y, cloglog(y, deriv=d, inverse=TRUE), col="chocolate", lwd=2)
    lines(y, cauchit(y, deriv=d, inverse=TRUE), col="tan", lwd=2)
    if(d==0) {
        abline(h=0.5, v=0, lty="dashed")
        legend(-4, 1, c("logit", "probit", "cloglog", "cauchit"),
               col=c("limegreen","purple","chocolate", "tan"), lwd=2)
    }
}
## End(Not run)
```

cauchy1 *Cauchy Distribution Family Function*

Description

Estimates the location parameter of the Cauchy distribution by maximum likelihood estimation.

Usage

```
cauchy1(scale.arg=1, llocation="identity",
        elocation=list(), ilocation=NULL, method.init=1)
```

Arguments

scale.arg	Known (positive) scale parameter, called s below.
llocation	Parameter link function for the a location parameter. See Links for more choices.
elocation	List. Extra argument for the link. See earg in Links for general information.
ilocation	Optional initial value for a. By default, an initial value is chosen internally.
method.init	Integer, either 1 or 2 or 3. Initial method, three algorithms are implemented. Choose the another value if convergence fails, or use ilocation.

Details

The Cauchy distribution has a density function

$$f(y; a, s) = \left\{\pi s[1 + ((y - a)/s)^2]\right\}^{-1}$$

where y and a are real and finite, and $s > 0$. The distribution is symmetric about a and has a heavy tail. Its median and mode are a, but the mean does not exist, therefore the fitted values are all NAs. Fisher scoring is used.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Note

Good initial values are needed. It pays to select a wide range of initial values via the ilocation and method.init arguments.

Author(s)

T. W. Yee

References

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

See Also

`Cauchy`, `cauchit`.

Examples

```
set.seed(123)
n = 500
x = runif(n)

y = rcauchy(n, loc=1+5*x, scale=.4)
fit = vglm(y ~ x, cauchy1(scale=0.4), trace =TRUE, crit="c")
coef(fit, matrix=TRUE)

y = rcauchy(n, loc=exp(1+0.5*x), scale=.4)
## Not run: hist(y)
fit = vglm(y ~ x, cauchy1(scale=0.4, lloc="loge"), trace=TRUE, crit="c")
coef(fit, matrix=TRUE)
fitted(fit)[1:4]
summary(fit)
```

---

ccoef-methods *Constrained (Canonical) Coefficients*

Description

`ccoef` is a generic function used to return the constrained (canonical) coefficients of a constrained ordination model. The function invokes particular methods which depend on the class of the first argument.

Methods

object The object from which the constrained coefficients are extracted.

`ccoef` *Extract Model Constrained/Canonical Coefficients*

Description

`ccoef` is a generic function which extracts the constrained (canonical) coefficients from objects returned by certain modelling functions.

Usage

```
ccoef(object, ...)
```

Arguments

`object`	An object for which the extraction of canonical coefficients is meaningful.
`...`	Other arguments fed into the specific methods function of the model.

Details

For constrained quadratic and ordination models, *canonical coefficients* are the elements of the **C** matrix used to form the latent variables. They are highly interpretable in ecology, and are looked at as weights or loadings.

They are also applicable for reduced-rank VGLMs.

Value

The value returned depends specifically on the methods function invoked.

Warning

For QO models, there is a direct inverse relationship between the scaling of the latent variables (site scores) and the tolerances. One normalization is for the latent variables to have unit variance. Another normalization is for all the species' tolerances to be unit (provided `EqualTolerances` is `TRUE`). These two normalizations cannot simultaneously hold in general. For rank R models with $R > 1$ it becomes more complicated because the latent variables are also uncorrelated. An important argument when fitting quadratic ordination models is whether `EqualTolerances` is `TRUE` or `FALSE`. See Yee (2004) for details.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

Yee, T. W. (2004) A new technique for maximum-likelihood canonical Gaussian ordination. *Ecological Monographs*, **74**, 685–701.

Yee, T. W. (2006) Constrained additive ordination. *Ecology*, **87**, 203–213.

See Also

ccoef-method, ccoef.qrrvglm, ccoef.cao, coef.

Examples

```
data(hspider)
set.seed(111)  # This leads to the global solution
hspider[,1:6] = scale(hspider[,1:6]) # Standardized environmental vars
p1 = cqo(cbind(Alopacce, Alopcune, Alopfabr, Arctlute, Arctperi,
               Auloalbi, Pardlugu, Pardmont, Pardnigr, Pardpull,
               Trocterr, Zoraspin) ~
         WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + ReflLux,
         fam = quasipoissonff, data = hspider, Crow1positive=FALSE)
ccoef(p1)
```

---

cdf.lmscreg *Cumulative Distribution Function for LMS Quantile Regression*

Description

Computes the cumulative distribution function (CDF) for observations, based on a LMS quantile regression.

Usage

```
cdf.lmscreg(object, newdata = NULL, ...)
```

Arguments

object	A **VGAM** quantile regression model, i.e., an object produced by modelling functions such as vglm and vgam with a family function beginning with "lms.".
newdata	Data frame where the predictions are to be made. If missing, the original data is used.
...	Parameters which are passed into functions such as cdf.lms.yjn.

Details

The CDFs returned here are values lying in [0,1] giving the relative probabilities associated with the quantiles newdata. For example, a value near 0.75 means it is close to the upper quartile of the distribution.

Value

A vector of CDF values lying in [0,1].

Note

The data are treated like quantiles, and the percentiles are returned. The opposite is performed by `qtplot.lmscreg`.

The CDF values of the model have been placed in `@post$cdf` when the model was fitted.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2004) Quantile regression via vector generalized additive models. *Statistics in Medicine*, **23**, 2295–2315.

Documentation accompanying the **VGAM** package at `http://www.stat.auckland.ac.nz/~yee` contains further information and examples.

See Also

`deplot.lmscreg`, `qtplot.lmscreg`, `lms.bcn`, `lms.bcg`, `lms.yjn`.

Examples

```
data(bminz)
fit = vgam(BMI ~ s(age, df=c(4,2)), fam=lms.bcn(zero=1), data=bminz)
fit@post$cdf[1:5]
cdf(fit)[1:5]   # Same
fit@y[1:5]
fitted(fit)[1:5,]

cdf(fit, data.frame(age=c(31.5,39), BMI=c(28.4,24)))
```

---

cgo *Redirects the user to cqo*

Description

Redirects the user to the function `cqo`.

Usage

```
cgo(...)
```

Arguments

...	Ignored.

Details

The former function cgo has been renamed cqo because CGO (for *canonical Gaussian ordination*) is a confusing and inaccurate name. CQO (for *constrained quadratic ordination*) is better. This new nomenclature described in Yee (2006).

Value

Nothing is returned; an error message is issued.

Warning

The code, therefore, in Yee (2004) will not run without changing the "g" to a "q".

Author(s)

Thomas W. Yee

References

Yee, T. W. (2004) A new technique for maximum-likelihood canonical Gaussian ordination. *Ecological Monographs*, **74**, 685–701.

Yee, T. W. (2006) Constrained additive ordination. *Ecology*, **87**, 203–213.

See Also

cqo.

Examples

```
## Not run:
cgo()
## End(Not run)
```

cgumbel *Censored Gumbel Distribution*

Description

Maximum likelihood estimation of the 2-parameter Gumbel distribution when there are censored observations. A matrix response is not allowed.

Usage

```
cgumbel(llocation="identity", lscale = "loge",
        elocation = list(), escale = list(), iscale=NULL,
        mean=TRUE, percentiles=NULL, zero=2)
```

Arguments

`llocation, lscale`
: Character. Parameter link functions for the location and (positive) *scale* parameters. See `Links` for more choices.

`elocation, escale`
: Extra argument for the respective links. See `earg` in `Links` for general information.

`iscale`
: Numeric and positive. Initial value for *scale*. Recycled to the appropriate length. In general, a larger value is better than a smaller value. The default is to choose the value internally.

`mean`
: Logical. Return the mean? If `TRUE` then the mean is returned, otherwise percentiles given by the `percentiles` argument.

`percentiles`
: Numeric with values between 0 and 100. If `mean=FALSE` then the fitted values are percentiles which must be specified by this argument.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The value (possibly values) must be from the set {1,2} corresponding respectively to *location* and *scale*. If `zero=NULL` then all linear/additive predictors are modelled as a linear combination of the explanatory variables. The default is to fit the shape parameter as an intercept only.

Details

This **VGAM** family function is like `gumbel` but handles observations that are left-censored (so that the true value would be less than the observed value) else right-censored (so that the true value would be greater than the observed value). To indicate which type of censoring, input `extra = list(leftcensored = vec1, rightcensored = vec2)` where `vec1` and `vec2` are logical vectors the same length as the response. If the two components of this list are missing then the logical values are taken to be `FALSE`. The fitted object has these two components stored in the `extra` slot.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Warning

Numerical problems may occur if the amount of censoring is excessive.

Note

See gumbel for details about the Gumbel distribution. The initial values are based on assuming all uncensored observations, therefore could be improved upon.

Author(s)

T. W. Yee

References

Coles, S. (2001) *An Introduction to Statistical Modeling of Extreme Values*. London: Springer-Verlag.

See Also

gumbel, egumbel, rgumbel, guplot, gev, venice.

Examples

```
# Example 1
data(venice)
ystar = venice[["r1"]]  # Use the first order statistic as the response
n = length(ystar)
L = runif(n, 100, 104) # Lower censoring points
U = runif(n, 130, 135) # Upper censoring points
y = pmax(L, ystar) # Left  censored
y = pmin(U, y)     # Right censored
extra = list(leftcensored = ystar < L, rightcensored = ystar > U)
fit = vglm(y ~ scale(year), data=venice, trace=TRUE, extra=extra,
           cgumbel(mean=FALSE, perc=c(5,25,50,75,95)))
coef(fit, matrix=TRUE)
fitted(fit)[1:5,]
fit@extra

# Example 2: simulated data
n = 1000
ystar = rgumbel(n, loc=1, scale=exp(0.5)) # The uncensored data
L = runif(n, -1, 1) # Lower censoring points
U = runif(n,  2, 5) # Upper censoring points
y = pmax(L, ystar) # Left  censored
y = pmin(U, y)     # Right censored
## Not run: par(mfrow=c(1,2)); hist(ystar); hist(y);
```

```
extra = list(leftcensored = ystar < L, rightcensored = ystar > U)
fit = vglm(y ~ 1, trace=TRUE, extra=extra, cgumbel)
coef(fit, matrix=TRUE)
```

`chest` *Chest Pain in NZ Adults*

Description

Presence/absence of chest pain in 10186 New Zealand adults.

Usage

```
data(chest)
```

Format

A data frame with 73 rows and the following 5 variables.

age a numeric vector; age (years).

nolnor a numeric vector of counts; no pain on LHS or RHS.

nolr a numeric vector of counts; no pain on LHS but pain on RHS.

lnor a numeric vector of counts; no pain on RHS but pain on LHS.

lr a numeric vector of counts; pain on LHS and RHS of chest.

Details

Each adult was asked their age and whether they experienced any pain or discomfort in their chest over the last six months. If yes, they indicated whether it was on their LHS and/or RHS of their chest.

Source

MacMahon, S., Norton, R., Jackson, R., Mackie, M. J., Cheng, A., Vander Hoorn, S., Milne, A., McCulloch, A. (1995) Fletcher Challenge-University of Auckland Heart & Health Study: design and baseline findings. *New Zealand Medical Journal*, **108**, 499–502.

Examples

```
data(chest)
fit = vgam(cbind(nolnor, nolr,lnor,lr) ~ s(age, c(4,3)),
           binom2.or(exchan=TRUE, zero=NULL), data = chest)
coef(fit, matrix=TRUE)
## Not run:
plot(fit, which.cf=2, se=TRUE)
## End(Not run)
```

chisq *Chi-squared Distribution*

Description

Maximum likelihood estimation of the degrees of freedom for a chi-squared distribution.

Usage

```
chisq(link = "loge", earg=list())
```

Arguments

link	Parameter link function. See Links for more choices.
earg	List. Extra argument for the link. See earg in Links for general information.

Details

The degrees of freedom is treated as a parameter to be estimated. It is treated as real and not integer. Being positive, a log link is used by default.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

There may be convergence problems if the degrees of freedom is very large.

Author(s)

T. W. Yee

References

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

See Also

Chisquare. normal1.

Examples

```
y = rchisq(n=200, df=exp(2))
fit = vglm(y ~ 1, chisq)
coef(fit, matrix=TRUE)
Coef(fit)
```

clo *Redirects the user to rrvglm*

Description

Redirects the user to the function rrvglm.

Usage

```
clo(...)
```

Arguments

...	Ignored.

Details

CLO stands for *constrained linear ordination*, and is fitted with a statistical class of models called *reduced-rank vector generalized linear models* (RR-VGLMs). It allows for generalized reduced-rank regression in that response types such as Poisson counts and presence/absence data can be handled.

Currently in the **VGAM** package, rrvglm is used to fit RR-VGLMs. However, the Author's opinion is that linear responses to a latent variable (composite environmental gradient) is not as common as unimodal responses, therefore cqo is often more appropriate.

The new CLO/CQO/CAO nomenclature described in Yee (2006).

Value

Nothing is returned; an error message is issued.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2006) Constrained additive ordination. *Ecology*, **87**, 203–213.

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

See Also

rrvglm, cqo.

Examples

```
## Not run:
clo()
## End(Not run)
```

`cloglog` *Complementary Log-log Link Function*

Description

Computes the complementary log-log transformation, including its inverse and the first two derivatives.

Usage

```
cloglog(theta, earg = list(), inverse = FALSE, deriv = 0,
        short = TRUE, tag = FALSE)
```

Arguments

`theta`	Numeric or character. See below for further details.
`earg`	Optional list. Extra argument for passing in additional information. Values of `theta` which are less than or equal to 0 can be replaced by the `bvalue` component of the list `earg` before computing the link function value. Values of `theta` which are greater than or equal to 1 can be replaced by 1 minus the `bvalue` component of the list `earg` before computing the link function value. The component name `bvalue` stands for "boundary value". See `Links` for general information about `earg`.
`inverse`	Logical. If `TRUE` the inverse function is computed.
`deriv`	Order of the derivative. Integer with value 0, 1 or 2.
`short`	Used for labelling the `blurb` slot of a `vglmff-class` object.
`tag`	Used for labelling the linear/additive predictor in the `initialize` slot of a `vglmff-class` object. Contains a little more information if `TRUE`.

Details

The complementary log-log link function is commonly used for parameters that lie in the unit interval. Numerical values of `theta` close to 0 or 1 or out of range result in `Inf`, `-Inf`, `NA` or `NaN`. The arguments `short` and `tag` are used only if `theta` is character.

Value

For `deriv = 0`, the complimentary log-log of `theta`, i.e., `log(-log(1 - theta))` when `inverse = FALSE`, and if `inverse = TRUE` then `1-exp(-exp(theta))`,.

For `deriv = 1`, then the function returns *d* `theta` / *d* `eta` as a function of `theta` if `inverse = FALSE`, else if `inverse = TRUE` then it returns the reciprocal.

Here, all logarithms are natural logarithms, i.e., to base e.

Note

Numerical instability may occur when `theta` is close to 1 or 0. One way of overcoming this is to use `earg`.

With constrained ordination (e.g., `cqo` and `cao`) used with `binomialff`, a complementary log-log link function is preferred over the default `logit` link, for a good reason. See the example below.

In terms of the threshold approach with cumulative probabilities for an ordinal response this link function corresponds to the extreme value distribution.

Author(s)

Thomas W. Yee

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

See Also

`Links`, `logit`, `probit`, `cauchit`.

Examples

```
p = seq(0.01, 0.99, by=0.01)
cloglog(p)
max(abs(cloglog(cloglog(p), inverse=TRUE) - p)) # Should be 0

p = c(seq(-0.02, 0.02, by=0.01), seq(0.97, 1.02, by=0.01))
cloglog(p)  # Has NAs
cloglog(p, earg=list(bvalue= .Machine$double.eps))  # Has no NAs

## Not run:
plot(p, logit(p), type="l", col="limegreen", ylab="transformation",
     lwd=2, las=1, main="Some probability link functions")
lines(p, probit(p), col="purple", lwd=2)
lines(p, cloglog(p), col="chocolate", lwd=2)
lines(p, cauchit(p), col="tan", lwd=2)
abline(v=0.5, h=0, lty="dashed")
legend(0.1, 4, c("logit", "probit", "cloglog", "cauchit"),
       col=c("limegreen","purple","chocolate", "tan"), lwd=2)
```

```
## End(Not run)

# This example shows that a cloglog link is preferred over the logit
n = 500; p = 5; S = 3; Rank = 1  # Species packing model:
mydata = rcqo(n, p, S, EqualTol=TRUE, ESOpt=TRUE, EqualMax=TRUE,
              family="binomial", hiabundance=5, seed=123, Rank=Rank)
fitc = cqo(attr(mydata, "formula"), ITol=TRUE, data=mydata,
           fam=binomialff(mv=TRUE, link="cloglog"), Rank=Rank)
fitl = cqo(attr(mydata, "formula"), ITol=TRUE, data=mydata,
           fam=binomialff(mv=TRUE, link="logit"), Rank=Rank)

# Compare the fitted models (cols 1 and 3) with the truth (col 2)
cbind(ccoef(fitc), attr(mydata, "ccoefficients"), ccoef(fitl))
```

`cnormal1` *Censored Normal Distribution*

Description

Maximum likelihood estimation for the normal distribution with left and right censoring.

Usage

```
cnormal1(lmu="identity", lsd="loge", imethod=1, zero=2)
```

Arguments

`lmu, lsd`	Parameter link functions applied to the mean and standard deviation parameters. See `Links` for more choices. The standard deviation is a positive quantity, therefore a log link is the default.
`imethod`	Initialization method. Either 1 or 2, this specifies two methods for obtaining initial values for the parameters.
`zero`	An integer vector, containing the value 1 or 2. If so, the mean or standard deviation respectively are modelled as an intercept only. Setting `zero=NULL` means both linear/additive predictors are modelled as functions of the explanatory variables.

Details

This function is like `normal1` but handles observations that are left-censored (so that the true value would be less than the observed value) else right-censored (so that the true value would be greater than the observed value). To indicate which type of censoring, input `extra = list(leftcensored = vec1, rightcensored = vec2)` where `vec1` and `vec2` are logical vectors the same length as the response. If the two components of this list are missing then the logical values are taken to be `FALSE`. The fitted object has these two components stored in the `extra` slot.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

This function was adapted from tobit. The Tobit model is a special case of this **VGAM** family function because the observations have a common lower censoring point and upper censoring point. If there are no censored observation then normal1 is recommended instead.

Author(s)

T. W. Yee

See Also

tobit, normal1, dcnormal1.

Examples

```
n = 1000
x = runif(n)
ystar  = rnorm(n, mean=100 + 15 * x, sd=exp(3)) # True values
## Not run: hist(ystar)
L = runif(n,  80,  90) # Lower censoring points
U = runif(n, 130, 140) # Upper censoring points
y = pmax(L, ystar) # Left  censored
y = pmin(U, y)     # Right censored
## Not run: hist(y)
extra = list(leftcensored = ystar < L, rightcensored = ystar > U)
fit = vglm(y ~ x, cnormal1(zero=2), trace=TRUE, extra=extra)
coef(fit, matrix=TRUE)
Coef(fit)
names(fit@extra)
```

---

coalminers *Breathlessness and Wheeze Amongst Coalminers*

Description

Coalminers who are smokers without radiological pneumoconiosis, classified by age, breathlessness and wheeze.

Usage

```
data(coalminers)
```

Format

A data frame with 9 age groups with the following 5 columns.

BW Counts with breathlessness and wheeze.

BnW Counts with breathlessness but no wheeze.

nBW Counts with no breathlessness but wheeze.

nBnW Counts with neither breathlessness or wheeze.

age Age of the coal miners (actually, the midpoints of the 5-year category ranges).

Details

The data were published in Ashford and Sowden (1970). A more recent analysis is McCullagh and Nelder (1989, Section 6.6).

Source

Ashford, J. R. and Sowden, R. R. (1970) Multi-variate probit analysis. *Biometrics*, **26**, 535–546.

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*. 2nd ed. London: Chapman & Hall.

Examples

```
data(coalminers)
str(coalminers)
```

---

constraints *Constraint Matrices*

Description

Returns the *constraint matrices* of objects in the **VGAM** package.

Usage

```
constraints(object, ...)
```

Arguments

`object`	Some **VGAM** object, for example, having class `vglmff-class`.
`...`	Other possible arguments.

Details

Constraint matrices describe the relationship of coefficients/component functions of a particular explanatory variable between the linear/additive predictors in VGLM/VGAM models. For example, they may be all different (constraint matrix is the identity matrix) or all the same (constraint matrix has one column and has unit values).

VGLMs and VGAMs have constraint matrices which are *known*. The class of RR-VGLMs have constraint matrices which are *unknown* and are to be estimated.

Value

This extractor function returns a list comprising of constraint matrices—one for each column of the LM model matrix, and in that order. The list is labelled with the variable names. Each constraint matrix has M rows, where M is the number of linear/additive predictors, and whose rank is equal to the number of columns. A model with no constraints at all has an order M identity matrix as each variable's constraint matrix.

Warning

The `xij` argument changes things, and this has not been fully resolved yet.

Note

In all **VGAM** family functions `zero=NULL` means none of the linear/additive predictors are modelled as intercepts-only. Other arguments found in certain **VGAM** family functions which affect constraint matrices include `parallel` and `exchangeable`.

The `constraints` argument in `vglm` and `vgam` allows constraint matrices to be inputted. If so, then `constraints(fit)` should return the same as the input.

Author(s)

T. W. Yee

References

Yee, T. W. and Wild, C. J. (1996) Vector generalized additive models. *Journal of the Royal Statistical Society, Series B, Methodological*, **58**, 481–493.

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

`http://www.stat.auckland.ac.nz/~yee` contains additional information.

See Also

VGLMs are described in `vglm-class`; RR-VGLMs are described in `rrvglm-class`.

Arguments such as `zero` and `parallel` found in many **VGAM** family functions are a way of creating/modifying constraint matrices conveniently, e.g., see `zero`.

Examples

```
# Fit the proportional odds model
data(pneumo)
pneumo = transform(pneumo, let=log(exposure.time))
fit = vglm(cbind(normal, mild, severe) ~ let,
           cumulative(parallel=TRUE, reverse=TRUE), pneumo)
fit
coef(fit, matrix=TRUE)
constraints(fit)  # Parallel assumption results in this



# Fit a rank-1 stereotype (RR-multinomial logit) model
data(car.all)
fit = rrvglm(Country ~ Width + Height + HP, multinomial, car.all, Rank=1)
constraints(fit) # All except the first are the A matrix
```

---

cqo *Fitting Constrained Quadratic Ordination (CQO)*

Description

A *constrained quadratic ordination* (CQO; formerly called *canonical Gaussian ordination* or CGO) model is fitted using the *quadratic reduced-rank vector generalized linear model* (QRR-VGLM) framework.

Usage

```
cqo(formula, family, data = list(), weights = NULL, subset = NULL,
    na.action = na.fail, etastart = NULL, mustart = NULL,
    coefstart = NULL, control = qrrvglm.control(...), offset = NULL,
    method = "cqo.fit", model = FALSE, x.arg = TRUE, y.arg = TRUE,
    contrasts = NULL, constraints = NULL, extra = NULL,
    smart = TRUE, ...)
```

Arguments

In this documentation, M is the number of linear predictors, S is the number of responses (species). Then $M = S$ for Poisson and binomial species data, and $M = 2S$ for negative binomial and gamma distributed species data.

`formula`	a symbolic description of the model to be fit. The RHS of the formula is applied to each linear predictor. Different variables in each linear predictor can be chosen by specifying constraint matrices.
`family`	a function of class `"vglmff"` describing what statistical model is to be fitted. Currently the following families are supported: `poissonff`, `binomialff` (`logit` and `cloglog` links available), `negbinomial`, `gamma2`, `gaussianff`. Sometimes special arguments are required for `cqo()`, e.g., `binomialff(mv=TRUE)`. Also, `quasipoissonff` and `quasibinomialff` may or may not work.

data	an optional data frame containing the variables in the model. By default the variables are taken from environment(formula), typically the environment from which cqo is called.
weights	an optional vector or matrix of (prior) weights to be used in the fitting process. Currently, this argument should not be used.
subset	an optional logical vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The "factory-fresh" default is na.omit.
etastart	starting values for the linear predictors. It is a M-column matrix. If $M = 1$ then it may be a vector. Currently, this argument probably should not be used.
mustart	starting values for the fitted values. It can be a vector or a matrix. Some family functions do not make use of this argument. Currently, this argument probably should not be used.
coefstart	starting values for the coefficient vector. Currently, this argument probably should not be used.
control	a list of parameters for controlling the fitting process. See qrrvglm.control for details.
offset	This argument must not be used.
method	the method to be used in fitting the model. The default (and presently only) method cqo.fit uses *iteratively reweighted least squares* (IRLS).
model	a logical value indicating whether the *model frame* should be assigned in the model slot.
x.arg, y.arg	logical values indicating whether the model matrix and response matrix used in the fitting process should be assigned in the x and y slots. Note the model matrix is the LM model matrix.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
constraints	an optional list of constraint matrices. The components of the list must be named with the term it corresponds to (and it must match in character format). Each constraint matrix must have M rows, and be of full-column rank. By default, constraint matrices are the M by M identity matrix unless arguments in the family function itself override these values. If constraints is used it must contain *all* the terms; an incomplete list is not accepted. Constraint matrices for x_2 variables are taken as the identity matrix.
extra	an optional list with any extra information that might be needed by the family function.
smart	logical value indicating whether smart prediction (smartpred) will be used.
...	further arguments passed into qrrvglm.control.

Details

QRR-VGLMs or *constrained quadratic ordination* (CQO) models are estimated here by maximum likelihood estimation. Optimal linear combinations of the environmental variables are computed,

called *latent variables* (these appear as `lv` for $R = 1$ else `lv1`, `lv2`, etc. in the output). Here, R is the *rank* or the number of ordination axes. Each species' response is then a regression of these latent variables using quadratic polynomials on a transformed scale (e.g., log for Poisson counts, logit for presence/absence responses). The solution is obtained iteratively in order to maximize the log-likelihood function, or equivalently, minimize the deviance.

The central formula (for Poisson and binomial species data) is given by

$$\eta = B_1^T x_1 + A\nu + \sum_{m=1}^{M} (\nu^T D_m \nu) e_m$$

where x_1 is a vector (usually just a 1 for an intercept), x_2 is a vector of environmental variables, $\nu = C^T x_2$ is a R-vector of latent variables, e_m is a vector of 0s but with a 1 in the mth position. The η are a vector of linear/additive predictors, e.g., the mth element is $\eta_m = \log(E[Y_m])$ for the mth species. The matrices B_1, A, C and D_m are estimated from the data, i.e., contain the regression coefficients. The tolerance matrices satisfy $T_s = -\frac{1}{2} D_s^{-1}$. Many important CQO details are directly related to arguments in `qrrvglm.control`, e.g., the argument `Norrr` specifies which variables comprise x_1.

Theoretically, the four most popular **VGAM** family functions to be used with `cqo` correspond to the Poisson, binomial, normal, and negative binomial distributions. The latter is a 2-parameter model. All of these are implemented, as well as the 2-parameter gamma. The Poisson is or should be catered for by `quasipoissonff` and `poissonff`, and the binomial by `quasibinomialff` and `binomialff`. Those beginning with `"quasi"` have dispersion parameters that are estimated for each species.

For initial values, the function `.Init.Poisson.QO` should work reasonably well if the data is Poisson with species having equal tolerances. It can be quite good on binary data too. Otherwise the `Cinit` argument in `qrrvglm.control` can be used.

It is possible to relax the quadratic form to an additive model. The result is a data-driven approach rather than a model-driven approach, so that CQO is extended to *constrained additive ordination* (CAO) when $R = 1$. See `cao` for more details.

Value

An object of class `"qrrvglm"`. Note that the slot `misc` has a list component called `deviance.Bestof` which gives the history of deviances over all the iterations.

Warning

Local solutions are not uncommon when fitting CQO models. To increase the chances of obtaining the global solution, increase the value of the argument `Bestof` in `qrrvglm.control`. For reproducibility of the results, it pays to set a different random number seed before calling `cqo` (the function `set.seed` does this). The function `cqo` chooses initial values for **C** using `.Init.Poisson.QO()` if `Use.Init.Poisson.QO=TRUE`, else random numbers.

Unless `ITolerances=TRUE` or `EqualTolerances=FALSE`, CQO is computationally expensive. It pays to keep the rank down to 1 or 2. If `EqualTolerances=TRUE` and `ITolerances=FALSE` then the cost grows quickly with the number of species and sites (in terms of memory requirements and time). The data needs to conform quite closely to the statistical model, and the environmental range of the data should be wide in order for the quadratics to fit the data well (bell-shaped response surfaces). If not, RR-VGLMs will be more appropriate because the response is linear on the

transformed scale (e.g., log or logit) and the ordination is called *constrained linear ordination* or CLO.

Like many regression models, CQO is sensitive to outliers (in the environmental and species data), sparse data, high leverage points, multicollinearity etc. For these reasons, it is necessary to examine the data carefully for these features and take corrective action (e.g., omitting certain species, sites, environmental variables from the analysis, transforming certain environmental variables, etc.). Any optimum lying outside the convex hull of the site scores should not be trusted. Fitting a CAO is recommended first, then upon transformations etc., possibly a CQO can be fitted.

For binary data, it is necessary to have 'enough' data. In general, the number of sites n ought to be much larger than the number of species S, e.g., at least 100 sites for two species. Compared to count (Poisson) data, numerical problems occur more frequently with presence/absence (binary) data. For example, if `Rank=1` and if the response data for each species is a string of all absences, then all presences, then all absences (when enumerated along the latent variable) then infinite parameter estimates will occur. In general, setting `ITolerances=TRUE` may help.

This function was formerly called `cgo`. It has been renamed to reinforce a new nomenclature described in Yee (2006).

Note

By default, a rank-1 equal-tolerances QRR-VGLM model is fitted (see `qrrvglm.control` for the default control parameters). The latent variables are always transformed so that they are uncorrelated. By default, the argument `trace` is `TRUE` meaning a running log is printed out while the computations are taking place. This is because the algorithm is computationally expensive, therefore users might think that their computers have frozen if `trace=FALSE`!

The argument `Bestof` in `qrrvglm.control` controls the number of models fitted (each uses different starting values) to the data. This argument is important because convergence may be to a *local* solution rather than the *global* solution. Using more starting values increases the chances of finding the global solution. Always plot an ordination diagram (use the generic function `lvplot`) and see if it looks sensible. Local solutions arise because the optimization problem is highly nonlinear, and this is particularly true for CAO.

Many of the arguments applicable to `cqo` are common to `vglm` and `rrvglm.control`. The most important arguments are `Rank`, `Norrr`, `Bestof`, `ITolerances`, `EqualTolerances`, `isdlv`, and `MUXfactor`.

When fitting a 2-parameter model such as the negative binomial or gamma, it pays to set `EqualTolerances=TRUE` and `ITolerances=FALSE`. This is because numerical problems can occur when fitting the model far away from the global solution when `ITolerances=TRUE`. Setting the two arguments as described will slow down the computation considerably, however it is numerically more stable.

In Example 1 below, an unequal-tolerances rank-1 QRR-VGLM is fitted to the hunting spiders dataset. In Example 2 below, an equal-tolerances rank-2 QRR-VGLM is fitted to the hunting spiders dataset. The numerical difficulties encountered in fitting the rank-2 model suggests a rank-1 model is probably preferable. In Example 3 below, constrained binary quadratic ordination (in old nomenclature, constrained Gaussian logit ordination) is fitted to some simulated data coming from a species packing model. With multivariate binary responses, one must use `mv=TRUE` to indicate that the response (matrix) is multivariate. Otherwise, it is interpreted as a single binary response variable.

Sometime in the future, this function might handle input of the form `cqo(x, y)`, where `x` and `y` are matrices containing the environmental and species data respectively.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2004) A new technique for maximum-likelihood canonical Gaussian ordination. *Ecological Monographs*, **74**, 685–701.

ter Braak, C. J. F. and Prentice, I. C. (1988) A theory of gradient analysis. *Advances in Ecological Research*, **18**, 271–317.

Yee, T. W. (2006) Constrained additive ordination. *Ecology*, **87**, 203–213.

See Also

qrrvglm.control, Coef.qrrvglm, rcqo, cao, uqo, rrvglm, poissonff, binomialff, negbinomial, gamma2, lvplot.qrrvglm, persp.qrrvglm, trplot.qrrvglm, vglm, set.seed, hspider.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

Examples

```
# Example 1; Fit an unequal tolerances model to the hunting spiders data
data(hspider)
hspider[,1:6]=scale(hspider[,1:6]) # Standardize the environmental variables
p1 = cqo(cbind(Alopacce, Alopcune, Alopfabr, Arctlute, Arctperi, Auloalbi,
               Pardlugu, Pardmont, Pardnigr, Pardpull, Trocterr, Zoraspin) ~
         WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + ReflLux,
         fam=poissonff, data=hspider, Crow1positive=FALSE, ITol=FALSE)
sort(p1@misc$deviance.Bestof) # A history of all the iterations
if(deviance(p1) > 1177) stop("suboptimal fit obtained")

## Not run:
S = ncol(p1@y) # Number of species
clr = (1:(S+1))[-7] # omits yellow
lvplot(p1, y=TRUE, lcol=clr, pch=1:S, pcol=clr, las=1) # ordination diagram
legend("topright", leg=dimnames(p1@y)[[2]], col=clr,
       pch=1:S, merge=TRUE, bty="n", lty=1:S, lwd=2)
## End(Not run)
(cp = Coef(p1))

(a = cp@lv[cp@lvOrder])  # The ordered site scores along the gradient
# Names of the ordered sites along the gradient:
rownames(cp@lv)[cp@lvOrder]
(a = (cp@Optimum)[,cp@OptimumOrder]) # The ordered optima along the gradient
a = a[!is.na(a)] # Delete the species that is not unimodal
names(a)         # Names of the ordered optima along the gradient

## Not run:
trplot(p1, whichSpecies=1:3, log="xy", type="b", lty=1, lwd=2,
       col=c("blue","red","green"), label=TRUE) -> ii # trajectory plot
```

```
legend(0.00005, 0.3, paste(ii$species[,1], ii$species[,2], sep=" and "),
       lwd=2, lty=1, col=c("blue","red","green"))
abline(a=0, b=1, lty="dashed")

S = ncol(p1@y) # Number of species
clr = (1:(S+1))[-7] # omits yellow
persp(p1, col=clr, label=TRUE, las=1) # perspective plot
## End(Not run)

# Example 2: A rank-2 equal tolerances CQO model with Poisson data
# This example is numerically fraught.
set.seed(555)
p2 = cqo(cbind(Alopacce, Alopcune, Alopfabr, Arctlute, Arctperi, Auloalbi,
               Pardlugu, Pardmont, Pardnigr, Pardpull, Trocterr, Zoraspin) ~
         WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + ReflLux,
         fam=poissonff, data=hspider, Crow1positive=FALSE,
#        ITol=FALSE, EqualTol=TRUE,
         Rank=2, Bestof=1, isdlv=c(2.1,0.9))
sort(p2@misc$deviance.Bestof) # A history of all the iterations
if(deviance(p2) > 1127) stop("suboptimal fit obtained")
## Not run:
lvplot(p2, ellips=FALSE, label=TRUE, xlim=c(-3,4),
       C=TRUE, Ccol="brown", sites=TRUE, scol="grey",
       pcol="blue", pch="+", chull=TRUE, ccol="grey")
## End(Not run)

# Example 3: species packing model with presence/absence data
n = 200; p = 5; S = 5
mydata = rcqo(n, p, S, fam="binomial", hiabundance=4,
              EqualTol=TRUE, ESOpt=TRUE, EqualMax=TRUE)
myform = attr(mydata, "formula")
b1 = cqo(myform, fam=binomialff(mv=TRUE, link="cloglog"), data=mydata)
sort(b1@misc$deviance.Bestof) # A history of all the iterations
## Not run:
lvplot(b1, y=TRUE, lcol=1:S, pch=1:S, pcol=1:S, las=1)
## End(Not run)
Coef(b1)

# Compare the fitted model with the 'truth'
cbind(truth=attr(mydata, "ccoefficients"), fitted=ccoef(b1))
```

`cratio` *Ordinal Regression with Continuation Ratios*

Description

Fits a continuation ratio logit/probit/cloglog/cauchit/... regression model to an ordered (preferably) factor response.

Usage

```
cratio(link = "logit", earg = list(),
       parallel = FALSE, reverse = FALSE, zero = NULL)
```

Arguments

In the following, the response Y is assumed to be a factor with ordered values $1, 2, \ldots, M+1$, so that M is the number of linear/additive predictors η_j.

link	Link function applied to the M continuation ratio probabilities. See Links for more choices.
earg	List. Extra argument for the link function. See earg in Links for general information.
parallel	A logical, or formula specifying which terms have equal/unequal coefficients.
reverse	Logical. By default, the continuation ratios used are $\eta_j = logit(P[Y > j \mid Y \geq j])$ for $j = 1, \ldots, M$. If reverse is TRUE, then $\eta_j = logit(P[Y < j+1 \mid Y \leq j+1])$ will be used.
zero	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,...,M}. The default value means none are modelled as intercept-only terms.

Details

There are a number of definitions for the *continuation ratio* in the literature. To make life easier, in the **VGAM** package, we use *continuation* ratios and *stopping* ratios (see sratio). Stopping ratios deal with quantities such as logit(P[Y=j|Y>=j]).

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Warning

No check is made to verify that the response is ordinal; see ordered.

Note

The response should be either a matrix of counts (with row sums that are all positive), or a factor. In both cases, the y slot returned by vglm/vgam/rrvglm is the matrix of counts.

For a nominal (unordered) factor response, the multinomial logit model (multinomial) is more appropriate.

Here is an example of the usage of the parallel argument. If there are covariates x1, x2 and x3, then parallel = TRUE ~ x1 + x2 -1 and parallel = FALSE ~ x3 are equivalent. This would constrain the regression coefficients for x1 and x2 to be equal; those of the intercepts and x3 would be different.

Author(s)

Thomas W. Yee

References

Agresti, A. (2002) *Categorical Data Analysis*, 2nd ed. New York: Wiley.

Simonoff, J. S. (2003) *Analyzing Categorical Data*, New York: Springer-Verlag.

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

sratio, acat, cumulative, multinomial, pneumo, logit, probit, cloglog, cauchit.

Examples

```
data(pneumo)
pneumo = transform(pneumo, let=log(exposure.time))
(fit = vglm(cbind(normal,mild,severe) ~ let, cratio(parallel=TRUE), pneumo))
coef(fit, matrix=TRUE)
constraints(fit)
predict(fit)
predict(fit, untransform=TRUE)
```

---

cumulative *Ordinal Regression with Cumulative Probabilities*

Description

Fits a cumulative logit/probit/cloglog/cauchit/... regression model to an ordered (preferably) factor response.

Usage

```
cumulative(link = "logit", earg = list(),
           parallel = FALSE, reverse = FALSE,
           mv = FALSE, intercept.apply = FALSE)
```

Arguments

In the following, the response Y is assumed to be a factor with ordered values $1, 2, \ldots, M+1$, so that M is the number of linear/additive predictors η_j.

link	Link function applied to the M cumulative probabilities. See Links for more choices.

`earg` List. Extra argument for the link function. See `earg` in `Links` for general information.

`parallel` A logical, or formula specifying which terms have equal/unequal coefficients.

`reverse` Logical. By default, the cumulative probabilities used are $P(Y \leq 1)$, $P(Y \leq 2)$, ..., $P(Y \leq M)$. If `reverse` is `TRUE`, then $P(Y \geq 2)$, $P(Y \geq 3)$, ..., $P(Y \geq M+1)$ will be used.

This should be set to `TRUE` for `link=` `golf`, `polf`, `nbolf`. For these links the cutpoints must be an increasing sequence; if `reverse=FALSE` for then the cutpoints must be an decreasing sequence.

`mv` Logical. Multivariate response? If `TRUE` then the input should be a matrix with values $1, 2, \ldots, L$, where L is the number of levels. Each column of the matrix is a response, i.e., multivariate response. A suitable matrix can be obtained from `Cut`.

`intercept.apply`
Logical. Whether the `parallel` argument should be applied to the intercept term. This should be set to `TRUE` for `link=` `golf`, `polf`, `nbolf`.

Details

By default, the non-parallel cumulative logit model is fitted, i.e.,

$$\eta_j = logit(P[Y \leq j])$$

where $j = 1, 2, \ldots, M$ and the η_j are not constrained to be parallel. This is also known as the *non-proportional odds model*. If the logit link is replaced by a complementary log-log link (`cloglog`) then this is known as the *proportional-hazards model*.

In almost all the literature, the constraint matrices associated with this family of models are known. For example, setting `parallel=TRUE` will make all constraint matrices (except for the intercept) equal to a vector of M 1's. If the constraint matrices are equal, unknown and to be estimated, then this can be achieved by fitting the model as a reduced-rank vector generalized linear model (RR-VGLM; see `rrvglm`). Currently, reduced-rank vector generalized additive models (RR-VGAMs) have not been implemented here.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

Warning

No check is made to verify that the response is ordinal; see `ordered`.

Note

The response should be either a matrix of counts (with row sums that are all positive), or a factor. In both cases, the `y` slot returned by `vglm`/`vgam`/`rrvglm` is the matrix of counts.

For a nominal (unordered) factor response, the multinomial logit model (`multinomial`) is more appropriate.

With the logit link, setting `parallel=TRUE` will fit a proportional odds model. Note that the `TRUE` here does not apply to the intercept term. In practice, the validity of the proportional odds assumption needs to be checked, e.g., by a likelihood ratio test. If acceptable on the data, then numerical problems are less likely to occur during the fitting, and there are less parameters. Numerical problems occur when the linear/additive predictors cross, which results in probabilities outside of (0,1); setting `parallel=TRUE` will help avoid this problem.

Here is an example of the usage of the `parallel` argument. If there are covariates `x1`, `x2` and `x3`, then `parallel = TRUE ~ x1 + x2 -1` and `parallel = FALSE ~ x3` are equivalent. This would constrain the regression coefficients for `x1` and `x2` to be equal; those of the intercepts and `x3` would be different.

In the future, this family function may be renamed to "`cups`" (for **cu**mulative **p**robabilitie**s**) or "`cute`" (for **cu**mulative probabili**t**i**e**s).

Author(s)

Thomas W. Yee

References

Agresti, A. (2002) *Categorical Data Analysis*, 2nd ed. New York: Wiley.

Dobson, A. J. (2001) *An Introduction to Generalized Linear Models*, 2nd ed. Boca Raton: Chapman & Hall/CRC Press.

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

Simonoff, J. S. (2003) *Analyzing Categorical Data*, New York: Springer-Verlag.

Yee, T. W. and Wild, C. J. (1996) Vector generalized additive models. *Journal of the Royal Statistical Society, Series B, Methodological*, **58**, 481–493.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

`acat`, `cratio`, `sratio`, `multinomial`, `pneumo`, `logit`, `probit`, `cloglog`, `cauchit`, `golf`, `polf`, `nbolf`.

Examples

```
# Fit the proportional odds model, p.179, in McCullagh and Nelder (1989)
data(pneumo)
pneumo = transform(pneumo, let=log(exposure.time))
(fit = vglm(cbind(normal, mild, severe) ~ let,
            cumulative(parallel=TRUE, reverse=TRUE), pneumo))
fit@y   # Sample proportions
weights(fit, type="prior")   # Number of observations
coef(fit, matrix=TRUE)
constraints(fit)   # Constraint matrices

# Check that the model is linear in let
```

```
fit2 = vgam(cbind(normal, mild, severe) ~ s(let, df=2),
            cumulative(reverse=TRUE), pneumo)
## Not run:
plot(fit2, se=TRUE, overlay=TRUE, lcol=1:2, scol=1:2)
## End(Not run)

# Check the proportional odds assumption with a likelihood ratio test
(fit3 = vglm(cbind(normal, mild, severe) ~ let,
             cumulative(parallel=FALSE, reverse=TRUE), pneumo))
1 - pchisq(2*(logLik(fit3)-logLik(fit)),
           df=length(coef(fit3))-length(coef(fit)))
```

`dagum` *Dagum Distribution Family Function*

Description

Maximum likelihood estimation of the 3-parameter Dagum distribution.

Usage

```
dagum(link.a = "loge", link.scale = "loge", link.p = "loge",
      earg.a=list(), earg.scale=list(), earg.p=list(),
      init.a = NULL, init.scale = NULL, init.p = 1, zero = NULL)
```

Arguments

`link.a, link.scale, link.p`
: Parameter link functions applied to the (positive) parameters `a`, `scale`, and `p`. See `Links` for more choices.

`earg.a, earg.scale, earg.p`
: List. Extra argument for each of the links. See `earg` in `Links` for general information.

`init.a, init.scale, init.p`
: Optional initial values for `a`, `scale`, and `p`.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. Here, the values must be from the set {1,2,3} which correspond to `a`, `scale`, `p`, respectively.

Details

The 3-parameter Dagum distribution is the 4-parameter generalized beta II distribution with shape parameter $q = 1$. It is known under various other names, such as the Burr III, inverse Burr, beta-K, and 3-parameter kappa distribution. It can be considered a generalized log-logistic distribution. Some distributions which are special cases of the 3-parameter Dagum are the inverse Lomax ($a = 1$), Fisk ($p = 1$), and the inverse paralogistic ($a = p$). More details can be found in Kleiber and Kotz (2003).

The Dagum distribution has a cumulative distribution function

$$F(y) = [1 + (y/b)^{-a}]^{-p}$$

which leads to a probability density function

$$f(y) = apy^{ap-1}/[b^{ap}\{1 + (y/b)^a\}^{p+1}]$$

for $a > 0$, $b > 0$, $p > 0$, $y > 0$. Here, b is the scale parameter `scale`, and the others are shape parameters. The mean is

$$E(Y) = b\,\Gamma(p + 1/a)\,\Gamma(1 - 1/a)/\Gamma(p)$$

provided $-ap < 1 < a$.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

If the self-starting initial values fail, try experimenting with the initial value arguments, especially those whose default value is not `NULL`.

From Kleiber and Kotz (2003), the MLE is rather sensitive to isolated observations located sufficiently far from the majority of the data. Reliable estimation of the scale parameter require $n > 7000$, while estimates for a and p can be considered unbiased for $n > 2000$ or 3000.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

See Also

`Dagum`, `genbetaII`, `betaII`, `sinmad`, `fisk`, `invlomax`, `lomax`, `paralogistic`, `invparalogistic`.

Examples

```
y = rdagum(n=3000, 4, 6, 2)
fit = vglm(y ~ 1, dagum, trace=TRUE)
fit = vglm(y ~ 1, dagum(init.a=2.1), trace=TRUE, crit="c")
coef(fit, mat=TRUE)
Coef(fit)
summary(fit)
```

dcnormal1 *Univariate Normal Distribution with Double Censoring*

Description

Maximum likelihood estimation of the two parameters of a univariate normal distribution when there is double censoring.

Usage

```
dcnormal1(r1 = 0, r2 = 0, link.sd = "loge",
          earg=list(), isd = NULL, zero = NULL)
```

Arguments

r1, r2	Integers. Number of smallest and largest values censored, respectively.
link.sd	Parameter link function applied to the standard deviation. See Links for more choices. Being a positive quantity, a log link is the default.
earg	List. Extra argument for the link. See earg in Links for general information.
isd	Numeric. Initial value for the standard deviation. The default value NULL means an initial value is obtained internally from the data.
zero	An integer with value 1 or 2. If so, the mean or standard deviation respectively are modelled as an intercept only. Usually, setting zero=2 will be used, if used at all. The default value NULL means both linear/additive predictors are modelled as functions of the explanatory variables.

Details

This family function uses the Fisher information matrix given in Harter and Moore (1966). The matrix is not diagonal if either r1 or r2 are positive.

By default, the mean is the first linear/additive predictor and the log of the standard deviation is the second linear/additive predictor.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

This family function only handles a vector or one-column matrix response. The weights argument, if used, are interpreted as frequencies, therefore it must be a vector with positive integer values.

With no censoring at all (the default), it is better (and equivalent) to use normal1.

Author(s)

T. W. Yee

References

Harter, H. L. and Moore, A. H. (1966) Iterative maximum-likelihood estimation of the parameters of normal populations from singly and doubly censored samples. *Biometrika*, **53**, 205–213.

See Also

normal1, tobit.

Examples

```
## Not run:
# Repeat the simulations described in Harter and Moore (1966)
SIMS = 100   # Number of simulations (change this to 1000)
mu.save = sd.save = rep(NA, len=SIMS)
r1 = 0; r2 = 4; n = 20
for(sim in 1:SIMS) {
    y = sort(rnorm(n))
    y = y[(1+r1):(n-r2)]  # Delete r1 smallest and r2 largest
    fit = vglm(y ~ 1, dcnormal1(r1=r1, r2=r2))
    mu.save[sim] = predict(fit)[1,1]
    sd.save[sim] = exp(predict(fit)[1,2])   # Assumes a log link and ~ 1
}
# Now look at the results
c(mean(mu.save), mean(sd.save))  # Should be c(0,1)
c(sd(mu.save), sd(sd.save))
## End(Not run)

# Data from Sarhan and Greenberg (1962); MLEs are mu=9.2606, sd=1.3754
strontium90 = c(8.2, 8.4, 9.1, 9.8, 9.9)
fit = vglm(strontium90 ~ 1, dcnormal1(r1=2, r2=3, isd=6), trace=TRUE)
coef(fit, matrix=TRUE)
Coef(fit)
```

---

deplot.lmscreg *Density Plot for LMS Quantile Regression*

Description

Plots a probability density function associated with a LMS quantile regression.

Usage

```
deplot.lmscreg(object, newdata = NULL,
               x0, y.arg, plot.it = TRUE, ...)
```

Arguments

object	A **VGAM** quantile regression model, i.e., an object produced by modelling functions such as vglm and vgam with a family function beginning with "lms.", e.g., lms.yjn.
newdata	Optional data frame containing secondary variables such as sex. It should have a maximum of one row. The default is to use the original data.
x0	Numeric. The value of the primary variable at which to make the 'slice'.
y.arg	Numerical vector. The values of the response variable at which to evaluate the density. This should be a grid that is fine enough to ensure the plotted curves are smooth.
plot.it	Logical. Plot it? If FALSE no plot will be done.
...	Graphical parameter that are passed into plotdeplot.lmscreg.

Details

This function calls, e.g., deplot.lms.yjn in order to compute the density function.

Value

The original object but with a list placed in the slot post, called @post$deplot. The list has components

newdata	The argument newdata above, or a one-row data frame constructed out of the x0 argument.
y	The argument y.arg above.
density	Vector of the density function values evaluated at y.arg.

Note

plotdeplot.lmscreg actually does the plotting.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2004) Quantile regression via vector generalized additive models. *Statistics in Medicine*, **23**, 2295–2315.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

plotdeplot.lmscreg, qtplot.lmscreg, lms.bcn, lms.bcg, lms.yjn.

Examples

```
## Not run:
data(bminz)
fit = vgam(BMI ~ s(age, df=c(4,2)), fam=lms.bcn(zero=1), data=bminz)
ygrid = seq(15, 43, by=0.25)
deplot(fit, x0=20, y=ygrid, xlab="BMI", col="green", llwd=2,
       main="BMI distribution at ages 20 (green), 40 (blue), 60 (red)")
deplot(fit, x0=40, y=ygrid, add=TRUE, col="blue", llwd=2)
deplot(fit, x0=60, y=ygrid, add=TRUE, col="red", llwd=2) -> a

names(a@post$deplot)
a@post$deplot$newdata
a@post$deplot$y[1:5]
a@post$deplot$density[1:5]
## End(Not run)
```

---

dexpbinomial *Double Exponential Binomial Distribution Family Function*

Description

Fits a double exponential binomial distribution by maximum likelihood estimation. The two parameters here are the mean and dispersion parameter.

Usage

```
dexpbinomial(lmean="logit", ldispersion="logit",
             emean=list(),  edispersion=list(),
             idispersion=0.25, zero=2)
```

Arguments

`lmean, ldispersion`
: Link functions applied to the two parameters, called μ and θ respectively below. See `Links` for more choices. The defaults cause the parameters to be restricted to $(0, 1)$.

`emean, edispersion`
: List. Extra argument for each of the links. See `earg` in `Links` for general information.

`idispersion`
: Initial value for the dispersion parameter. If given, it must be in range, and is recyled to the necessary length. Use this argument if convergence failure occurs.

`zero`
: An integer specifying which linear/additive predictor is to be modelled as an intercept only. If assigned, the single value should be either `1` or `2`. The default is to have a single dispersion parameter value. To model both parameters as functions of the covariates assign `zero=NULL`.

Details

This distribution provides a way for handling overdispersion in a binary response. The double exponential binomial distribution belongs the family of double exponential distributions proposed by Efron (1986). Below, equation numbers refer to that original article. Briefly, the idea is that an ordinary one-parameter exponential family allows the addition of a second parameter θ which varies the dispersion of the family without changing the mean. The extended family behaves like the original family with sample size changed from n to $n\theta$. The extended family is an exponential family in μ when n and θ are fixed, and an exponential family in θ when n and μ are fixed. Having $0 < \theta < 1$ corresponds to overdispersion with respect to the binomial distribution. See Efron (1986) for full details.

This **VGAM** family function implements an *approximation* (2.10) to the exact density (2.4). It replaces the normalizing constant by unity since the true value nearly equals 1. The default model fitted is $\eta_1 = logit(\mu)$ and $\eta_2 = logit(\theta)$. This restricts both parameters to lie between 0 and 1, although the dispersion parameter can be modelled over a larger parameter space by assigning the arguments `ldispersion` and `edispersion`.

Approximately, the mean (of Y) is μ. The *effective sample size* is the dispersion parameter multiplied by the original sample size, i.e., $n\theta$. This family function uses Fisher scoring, and the two estimates are asymptotically independent because the expected information matrix is diagonal.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`.

Warning

Numerical difficulties can occur; if so, try using `idispersion`.

Note

This function processes the input in the same way as `binomialff`, however multivariate responses are not allowed (`binomialff(mv=FALSE)`).

Author(s)

T. W. Yee

References

Efron, B. (1986) Double exponential families and their use in generalized linear regression. *Journal of the American Statistical Association*, **81**, 709–721.

See Also

`binomialff`, `toxop`.

Examples

```
# This example mimics the example in Efron (1986). The results here
# differ slightly.
data(toxop)

# Scale the variables
toxop = transform(toxop,
                  phat = positive / ssize,
                  srainfall = scale(rainfall),  # (6.1)
                  sN = scale(ssize))            # (6.2)

# A fit similar (should be identical) to Section 6 of Efron (1986).
# But does not use poly(), and M=1.25 here, as in (5.3)
cmlist = list("(Intercept)"=diag(2),
              "I(srainfall)"=rbind(1,0),
              "I(srainfall^2)"=rbind(1,0),
              "I(srainfall^3)"=rbind(1,0),
              "I(sN)"=rbind(0,1),
              "I(sN^2)"=rbind(0,1))
elist = list(min=0, max=1.25)
fit = vglm(phat ~ I(srainfall) + I(srainfall^2) + I(srainfall^3) +
                  I(sN) + I(sN^2),
           fam = dexpbinomial(ldisp="elogit", idisp=0.2,
                              edisp=elist, zero=NULL),
           data=toxop, weight=ssize, trace=TRUE, constraints=cmlist)

# Now look at the results
coef(fit)
coef(fit, matrix=TRUE)
fitted(fit)[1:4,]
summary(fit)
vcov(fit)
sqrt(diag(vcov(fit)))   # Standard errors

# Effective sample size (not quite the last column of Table 1)
predict(fit)[1:4,]
Dispersion = elogit(predict(fit)[,2], earg=elist, inverse=TRUE)
c(round(weights(fit, type="prior") * Dispersion, dig=1))

# Ordinary logistic regression (gives same results as (6.5))
ofit = vglm(phat ~ I(srainfall) + I(srainfall^2) + I(srainfall^3),
            fam = binomialff, data=toxop, weight=ssize, trace=TRUE)

# Same as fit but it uses poly(), and can be plotted (cf. Figure 1)
cmlist2 = list("(Intercept)"=diag(2),
               "poly(srainfall, 3)"=rbind(1,0),
               "poly(sN, 2)"=rbind(0,1))
fit2 = vglm(phat ~ poly(srainfall, 3) + poly(sN, 2),
            fam = dexpbinomial(ldisp="elogit", idisp=0.2,
                               edisp=list(min=0, max=1.25), zero=NULL),
            data=toxop, weight=ssize, trace=TRUE, constraints=cmlist2)
## Not run:
```

```
par(mfrow=c(1,2))
plotvgam(fit2, se=TRUE, lcol="blue", scol="red")  # Cf. Figure 1

# Cf. Figure 1(a)
par(mfrow=c(1,2))
o = with(toxop, sort.list(rainfall))
with(toxop, plot(rainfall[o], fitted(fit2)[o], type="l", col="blue",
                 las=1, ylim=c(0.3, 0.65)))
with(toxop, points(rainfall[o], fitted(ofit)[o], col="red", type="b",
                   pch=19))

# Cf. Figure 1(b)
o = with(toxop, sort.list(ssize))
with(toxop, plot(ssize[o], Dispersion[o], type="l", col="blue", las=1,
                 xlim=c(0, 100)))
## End(Not run)
```

`dirichlet` *Fitting a Dirichlet Distribution*

Description

Fits a Dirichlet distribution to a matrix of compositions.

Usage

```
dirichlet(link = "loge", earg=list(), zero=NULL)
```

Arguments

In the following, the response is assumed to be a M-column matrix with positive values and whose rows each sum to unity. Such data can be thought of as compositional data. There are M linear/additive predictors η_j.

`link`	Link function applied to each of the M (positive) shape parameters α_j. See `Links` for more choices. The default gives $\eta_j = \log(\alpha_j)$.
`earg`	List. Extra argument for the link. See `earg` in `Links` for general information.
`zero`	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The default is none of them. If used, choose values from the set $\{1, 2, \ldots, M\}$.

Details

The Dirichlet distribution is commonly used to model compositional data, including applications in genetics. Suppose $(Y_1, \ldots, Y_M)^T$ is the response. Then it has a Dirichlet distribution if $(Y_1, \ldots, Y_{M-1})^T$ has density

$$\frac{\Gamma(\alpha_+)}{\prod_{j=1}^{M} \Gamma(\alpha_j)} \prod_{j=1}^{M} y_j^{\alpha_j - 1}$$

where $\alpha_+ = \alpha_1 + \cdots + \alpha_M$, $\alpha_j > 0$, and the density is defined on the unit simplex

$$\Delta_M = \left\{ (y_1, \ldots, y_M)^T : y_1 > 0, \ldots, y_M > 0, \sum_{j=1}^{M} y_j = 1 \right\}.$$

One has $E(Y_j) = \alpha_j/\alpha_+$, which are returned as the fitted values. For this distribution Fisher scoring corresponds to Newton-Raphson.

The Dirichlet distribution can be motivated by considering the random variables $(G_1, \ldots, G_M)^T$ which are each independent and identically distributed as a gamma distribution with density $f(g_j) = g_j^{\alpha_j - 1} e^{-g_j}/\Gamma(\alpha_j)$. Then the Dirichlet distribution arises when $Y_j = G_j/(G_1 + \cdots + G_M)$.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

When fitted, the `fitted.values` slot of the object contains the M-column matrix of means.

Note

The response should be a matrix of positive values whose rows each sum to unity. Similar to this is count data, where probably a multinomial logit model (`multinomial`) may be appropriate. Another similar distribution to the Dirichlet is the Dirichlet-multinomial (see `dirmultinomial`).

Author(s)

Thomas W. Yee

References

Lange, K. (2002) *Mathematical and Statistical Methods for Genetic Analysis*, 2nd ed. New York: Springer-Verlag.

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

See Also

`rdiric`, `dirmultinomial`, `multinomial`.

Examples

```
y = rdiric(n=1000, shape=c(3, 1, 4))
fit = vglm(y ~ 1, dirichlet, trace = TRUE, crit="c")
Coef(fit)
coef(fit, matrix=TRUE)
fitted(fit)[1:2,]
```

`dirmul.old` *Fitting a Dirichlet-Multinomial Distribution*

Description

Fits a Dirichlet-multinomial distribution to a matrix of non-negative integers.

Usage

```
dirmul.old(link = "loge", earg = list(),
           init.alpha = 0.01, parallel = FALSE, zero = NULL)
```

Arguments

`link`	Link function applied to each of the M (positive) shape parameters α_j for $j = 1, \ldots, M$. See `Links` for more choices. Here, M is the number of columns of the response matrix.
`earg`	List. Extra argument for `link`. See `earg` in `Links` for general information.
`init.alpha`	Numeric vector. Initial values for the `alpha` vector. Must be positive. Recycled to length M.
`parallel`	A logical, or formula specifying which terms have equal/unequal coefficients.
`zero`	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,...,M}.

Details

The Dirichlet-multinomial distribution, which is somewhat similar to a Dirichlet distribution, has probability function

$$P(Y_1 = y_1, \ldots, Y_M = y_M) = \binom{2y_*}{y_1, \ldots, y_M} \frac{\Gamma(\alpha_+)}{\Gamma(2y_* + \alpha_+)} \prod_{j=1}^{M} \frac{\Gamma(y_j + \alpha_j)}{\Gamma(\alpha_j)}$$

for $\alpha_j > 0$, $\alpha_+ = \alpha_1 + \cdots + \alpha_M$, and $2y_* = y_1 + \cdots + y_M$. Here, $\binom{a}{b}$ means "a choose b" and refers to combinations (see `choose`). The (posterior) mean is

$$E(Y_j) = (y_j + \alpha_j)/(2y_* + \alpha_+)$$

for $j = 1, \ldots, M$, and these are returned as the fitted values as a M-column matrix.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

Note

The response should be a matrix of non-negative values. Convergence seems to slow down if there are zero values. Currently, initial values can be improved upon.

This function is almost defunct and may be withdrawn soon. Use `dirmultinomial` instead.

Author(s)

Thomas W. Yee

References

Lange, K. (2002) *Mathematical and Statistical Methods for Genetic Analysis*, 2nd ed. New York: Springer-Verlag.

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

Paul, S. R., Balasooriya, U. and Banerjee, T. (2005) Fisher information matrix of the Dirichlet-multinomial distribution. *Biometrical Journal*, **47**, 230–236.

Documentation accompanying the **VGAM** package at `http://www.stat.auckland.ac.nz/~yee` contains further information and examples.

See Also

`dirmultinomial`, `dirichlet`, `betabin.ab`, `multinomial`.

Examples

```
# Data from p.50 of Lange (2002)
alleleCounts = c(2, 84, 59, 41, 53, 131, 2, 0,
       0, 50, 137, 78, 54, 51, 0, 0,
       0, 80, 128, 26, 55, 95, 0, 0,
       0, 16, 40, 8, 68, 14, 7, 1)
dim(alleleCounts) = c(8, 4)
alleleCounts = data.frame(t(alleleCounts))
dimnames(alleleCounts) = list(c("White","Black","Chicano","Asian"),
                    paste("Allele", 5:12, sep=""))

set.seed(123)  # @initialize uses random numbers
fit = vglm(cbind(Allele5,Allele6,Allele7,Allele8,Allele9,
                 Allele10,Allele11,Allele12) ~ 1, dirmul.old,
            trace=TRUE, crit="c", data=alleleCounts)

(sfit = summary(fit))
vcov(sfit)
round(eta2theta(coef(fit), fit@misc$link), dig=2)  # not preferred
round(Coef(fit), dig=2) # preferred  # preferred
round(t(fitted(fit)), dig=4) # 2nd row of Table 3.5 of Lange (2002)
coef(fit, matrix=TRUE)

pfit = vglm(cbind(Allele5,Allele6,Allele7,Allele8,Allele9,
```

```
                  Allele10,Allele11,Allele12) ~ 1,
              dirmul.old(parallel=TRUE), trace=TRUE,
              data=alleleCounts)
round(eta2theta(coef(pfit), pfit@misc$link), dig=2) # not preferred
round(Coef(pfit), dig=2)   # preferred
```

`dirmultinomial` *Fitting a Dirichlet-Multinomial Distribution*

Description

Fits a Dirichlet-multinomial distribution to a matrix response.

Usage

```
dirmultinomial(lphi="logit", ephi = list(), iphi = 0.10,
               parallel= FALSE, zero="M")
```

Arguments

`lphi`	Link function applied to the ϕ parameter, which lies in the open unit interval $(0, 1)$. See `Links` for more choices.
`ephi`	List. Extra argument for `lphi`. See `earg` in `Links` for general information.
`iphi`	Numeric. Initial value for ϕ. Must be in the open unit interval $(0, 1)$. If a failure to converge occurs try assigning this argument a different value.
`parallel`	A logical (formula not allowed here) indicating whether the probabilities $\pi_1, \ldots, \pi_{M-1}$ are to be equal via equal coefficients. Note π_M will generally be different from the other probabilities. Setting `parallel=TRUE` will only work if you also set `zero=NULL` because of interference between these arguments (with respect to the intercept term).
`zero`	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set $\{1, 2, \ldots, M\}$. If the character `"M"` then this means the numerical value M, which corresponds to linear/additive predictor associated with ϕ. Setting `zero=NULL` means none of the values from the set $\{1, 2, \ldots, M\}$.

Details

The Dirichlet-multinomial distribution arises from a multinomial distribution where the probability parameters are not constant but are generated from a multivariate distribution called the Dirichlet distribution. The Dirichlet-multinomial distribution has probability function

$$P(Y_1 = y_1, \ldots, Y_M = y_M) = \binom{N_*}{y_1, \ldots, y_M} \frac{\prod_{j=1}^{M} \prod_{r=1}^{y_j} (\pi_j(1-\phi) + (r-1)\phi)}{\prod_{r=1}^{N_*} (1 - \phi + (r-1)\phi)}$$

where ϕ is the *over-dispersion* parameter and $N_* = y_1 + \cdots + y_M$. Here, $\binom{a}{b}$ means "a choose b" and refers to combinations (see `choose`). The above formula applies to each row of the matrix

response. In this **VGAM** family function the first $M-1$ linear/additive predictors correspond to the first $M-1$ probabilities via

$$\eta_j = \log(P[Y=j]/P[Y=M]) = \log(\pi_j/\pi_M)$$

where η_j is the jth linear/additive predictor ($\eta_M = 0$ by definition for $P[Y=M]$ but not for ϕ) and $j = 1, \ldots, M-1$. The Mth linear/additive predictor corresponds to `lphi` applied to ϕ.

Note that $E(Y_j) = N_* \pi_j$ but the probabilities (returned as the fitted values) π_j are bundled together as a M-column matrix. The quantities N_* are returned as the prior weights.

The beta-binomial distribution is a special case of the Dirichlet-multinomial distribution when $M = 2$; see `betabinomial`. It is easy to show that the first shape parameter of the beta distribution is $shape1 = \pi(1/\phi - 1)$ and the second shape parameter is $shape2 = (1-\pi)(1/\phi - 1)$. Also, $\phi = 1/(1 + shape1 + shape2)$, which is known as the *intra-cluster correlation* coefficient.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

If the model is an intercept-only model then `@misc` (which is a list) has a component called `shape` which is a vector with the M values $\pi_j(1/\phi - 1)$.

Warning

This **VGAM** family function is prone to numerical problems, especially when there are covariates.

Note

The response can be a matrix of non-negative integers, or else a matrix of sample proportions and the total number of counts in each row specified using the `weights` argument. This dual input option is similar to `multinomial`.

To fit a 'parallel' model with the ϕ parameter being an intercept-only you will need to use the `constraints` argument.

Currently, Fisher scoring is implemented. To compute the expected information matrix a `for` loop is used; this may be very slow when the counts are large. Additionally, convergence may be slower than usual due to round-off error when computing the expected information matrices.

Author(s)

Thomas W. Yee

References

Paul, S. R., Balasooriya, U. and Banerjee, T. (2005) Fisher information matrix of the Dirichlet-multinomial distribution. *Biometrical Journal*, **47**, 230–236.

See Also

`dirmul.old`, `betabinomial`, `betabin.ab`, `dirichlet`, `multinomial`.

Examples

```
n = 10
M = 5
y = round(matrix(runif(n*M)*10, n, M))  # Integer counts
fit = vglm(y ~ 1, dirmultinomial, trace=TRUE)
fitted(fit)[1:2,]
fit@y  # Sample proportions
weights(fit, type="prior", matrix=FALSE) # Total counts per row

x = runif(n)
fit = vglm(y ~ x, dirmultinomial, trace=TRUE)
## Not run:
Coef(fit)   # This does not work
## End(Not run)
coef(fit, matrix=TRUE)
(sfit = summary(fit))
vcov(sfit)
```

Zeta *The Zeta Distribution*

Description

Density for the zeta distribution.

Usage

```
dzeta(x, p)
```

Arguments

x	Numerical vector/matrix to evaluate the density.
p	The parameter p. This must be greater than 1.

Details

The density function of the zeta distribution is given by

$$y^{-p}/\zeta(p)$$

where $p > 1$, $y = 1, 2, \ldots$, and ζ is Riemann's zeta function.

Value

Returns the density evaluated at x.

Warning

This function has not been fully tested.

Note

The **VGAM** family function `zetaff` estimates the parameter p.

Author(s)

T. W. Yee

References

Johnson N. L., Kotz S., and Balakrishnan N. (1993) *Univariate Discrete Distributions*, 2nd ed. New York: Wiley.

See Also

`zeta`, `zetaff`.

Examples

```
dzeta(1:20, p=2)
## Not run:
plot(1:6, dzeta(1:6, p=4), type="h", las=1, ylab="Probability",
     main="zeta probability function; black: p = 4; blue: p = 2")
points(0.10+1:6, dzeta(1:6, p=2), type="h", col="blue")
## End(Not run)
```

---

enzyme *Enzyme Data*

Description

Enzyme velocity and substrate concentration.

Usage

```
data(enzyme)
```

Format

A data frame with 12 observations on the following 2 variables.

conc a numeric explanatory vector; substrate concentration

velocity a numeric response vector; enzyme velocity

Details

Sorry, more details need to be included later.

Source

Sorry, more details need to be included later.

References

Watts, D. G. (1981) An introduction to nonlinear least squares. In: L. Endrenyi (Ed.), *Kinetic Data Analysis: Design and Analysis of Enzyme and Pharmacokinetic Experiments*, pp.1–24. New York: Plenum Press.

See Also

`micmen`.

Examples

```
data(enzyme)
fit = vglm(velocity ~ 1, micmen, enzyme, trace = TRUE, crit = "c",
           regressor = enzyme$conc)
## Not run:
attach(enzyme)
plot(conc, velocity, xlab="concentration", las=1, main="enzyme data")
lines(conc, fitted(fit), col="blue")
detach(enzyme)
## End(Not run)

summary(fit)
```

`erf` *Error Function*

Description

Computes the error function based on the normal distribution.

Usage

```
erf(x)
```

Arguments

`x`	Numeric.

Details

$Erf(x)$ is defined as

$$Erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2) dt$$

so that it is closely related to `pnorm`.

Value

Returns the value of the function evaluated at `x`.

Note

Some authors omit the term $2/\sqrt{\pi}$ from the definition of $Erf(x)$. Although defined for complex arguments, this function only works for real arguments.

The *complementary error function* $erfc(x)$ is defined as $1 - erf(x)$, and is implemented by `erfc`.

Author(s)

T. W. Yee

References

Abramowitz, M. and Stegun, I. A. (1972) *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, New York: Dover Publications Inc.

See Also

`pnorm`.

Examples

```
## Not run:
x = seq(-3, 3, len=200)
plot(x, erf(x), type="l", col="red", las=1, lwd=2,
     main="red is erf(x), blue is pnorm(x)")
abline(v=0, h=0, lty="dashed", col="black")
lines(x, pnorm(x), col="blue", lty="dotted", lwd=2)
## End(Not run)
```

---

erlang *Erlang Distribution*

Description

Estimates the scale parameter of the Erlang distribution by maximum likelihood estimation.

Usage

```
erlang(shape.arg, link = "loge", earg=list(), method.init = 1)
```

Arguments

shape.arg	The shape parameter. The user must specify a positive integer.
link	Link function applied to the (positive) $scale$ parameter. See Links for more choices.
earg	List. Extra argument for the link. See earg in Links for general information.
method.init	An integer with value 1 or 2 which specifies the initialization method. If failure to converge occurs try the other value.

Details

The Erlang distribution is a special case of the gamma distribution with *shape* that is a positive integer. If shape.arg=1 then it simplifies to the exponential distribution. As illustrated in the example below, the Erlang distribution is the distribution of the sum of shape.arg independent and identically distributed exponential random variates.

The probability density function of the Erlang distribution is given by

$$f(y) = \exp(-y/scale) y^{shape-1} scale^{-shape} / \Gamma(shape)$$

for known positive integer $shape$, unknown $scale > 0$ and $y > 0$. Here, $\Gamma(shape)$ is the gamma function, as in gamma. The mean of Y is $\mu = shape \times scale$ and its variance is $shape \times scale^2$. The linear/additive predictor, by default, is $\eta = \log(scale)$.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

The rate parameter found in gamma2.ab is 1/scale here—see also rgamma.

Author(s)

T. W. Yee

References

Most standard texts on statistical distributions describe this distribution, e.g.,

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

See Also

gamma2.ab, exponential.

Examples

```
n = 200; rate = 8
y = rep(0, n)
for(i in 1:3)
    y = y + rexp(n, rate=rate)
fit = vglm(y ~ 1, erlang(shape=3), trace=TRUE)
coef(fit, matrix=TRUE)
Coef(fit) # Answer = 1/rate
1/rate
summary(fit)
```

`expexp` *Exponentiated Exponential Distribution*

Description

Estimates the two parameters of the exponentiated exponential distribution by maximum likelihood estimation.

Usage

```
expexp(lshape = "loge", lscale = "loge",
       eshape=list(), escale=list(),
       ishape = 1.1, iscale = NULL,
       tolerance = 1.0e-6, zero = NULL)
```

Arguments

`lshape, lscale`
: Parameter link functions for the α and λ parameters. See `Links` for more choices. The defaults ensure both parameters are positive.

`eshape, escale`
: List. Extra argument for each of the links. See `earg` in `Links` for general information.

`ishape`
: Initial value for the α parameter. If convergence fails try setting a different value for this argument.

`iscale`
: Initial value for the λ parameter. By default, an initial value is chosen internally using `ishape`.

`tolerance`
: Numeric. Small positive value for testing whether values are close enough to 1 and 2.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The default is none of them. If used, choose one value from the set {1,2}.

Details

The exponentiated exponential distribution is an alternative to the Weibull and the gamma distributions. The formula for the density is

$$f(y;\alpha,\lambda) = \alpha\lambda(1-\exp(-\lambda y))^{\alpha-1}\exp(-\lambda y)$$

where $y > 0$, $\alpha > 0$ and $\lambda > 0$. The mean of Y is $(\psi(\alpha+1) - \psi(1))/\lambda$ (returned as the fitted values) where ψ is the digamma function. The variance of Y is $(\psi'(1) - \psi'(\alpha+1))/\lambda^2$ where ψ' is the trigamma function.

This distribution has been called the two-parameter generalized exponential distribution by Gupta and Kundu (2006). A special case of the exponentiated exponential distribution: $\alpha = 1$ is the exponential distribution.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Warning

Practical experience shows that reasonably good initial values really helps. In particular, try setting different values for the `ishape` argument if numerical problems are encountered or failure to convergence occurs. Even if convergence occurs try perturbing the initial value to make sure the global solution is obtained and not a local solution. The algorithm may fail if the estimate of the shape parameter is too close to unity.

Note

Fisher scoring is used, however, convergence is usually very slow. This is a good sign that there is a bug, but I have yet to check that the expected information is correct. Also, I have yet to implement Type-I right censored data using the results of Gupta and Kundu (2006).

Another algorithm for fitting this model is implemented in `expexp1`.

Author(s)

T. W. Yee

References

Gupta, R. D. and Kundu, D. (2001) Exponentiated exponential family: an alternative to gamma and Weibull distributions, *Biometrical Journal*, **43**, 117–130.

Gupta, R. D. and Kundu, D. (2006) On the comparison of Fisher information of the Weibull and GE distributions, *Journal of Statistical Planning and Inference*, **136**, 3130–3144.

See Also

`expexp1`, `gamma2.ab`, `weibull`.

Examples

```
# A special case: exponential data
y = rexp(n <- 1000)
fit = vglm(y ~ 1, fam=expexp, trace=TRUE, maxit=99)
coef(fit, matrix=TRUE)
Coef(fit)

# Ball bearings data (number of million revolutions before failure)
bbearings = c(17.88, 28.92, 33.00, 41.52, 42.12, 45.60,
48.80, 51.84, 51.96, 54.12, 55.56, 67.80, 68.64, 68.64,
68.88, 84.12, 93.12, 98.64, 105.12, 105.84, 127.92,
128.04, 173.40)
fit = vglm(bbearings ~ 1, fam=expexp(iscale=0.05, ish=5),
           trace=TRUE, maxit=300)
coef(fit, matrix=TRUE)
Coef(fit)   # Authors get c(shape=5.2589, scale=0.0314)
logLik(fit) # Authors get -112.9763

# Failure times of the airconditioning system of an airplane
acplane = c(23, 261, 87, 7, 120, 14, 62, 47,
225, 71, 246, 21, 42, 20, 5, 12, 120, 11, 3, 14,
71, 11, 14, 11, 16, 90, 1, 16, 52, 95)
fit = vglm(acplane ~ 1, fam=expexp(ishape=0.8, isc=0.15),
           trace=TRUE, maxit=99)
coef(fit, matrix=TRUE)
Coef(fit)   # Authors get c(shape=0.8130, scale=0.0145)
logLik(fit) # Authors get log-lik -152.264
```

`expexp1` *Exponentiated Exponential Distribution*

Description

Estimates the two parameters of the exponentiated exponential distribution by maximizing a profile (concentrated) likelihood.

Usage

```
expexp1(lscale = "loge", escale=list(), iscale = NULL, ishape = 1)
```

Arguments

`lscale`	Parameter link function for the (positive) λ parameter. See `Links` for more choices.
`escale`	List. Extra argument for the link. See `earg` in `Links` for general information.
`iscale`	Initial value for the λ parameter. By default, an initial value is chosen internally using `ishape`.
`ishape`	Initial value for the α parameter. If convergence fails try setting a different value for this argument.

Details

See expexp for details about the exponentiated exponential distribution. This family function uses a different algorithm for fitting the model. Given λ, the MLE of α can easily be solved in terms of λ. This family function maximizes a profile (concentrated) likelihood with respect to λ. Newton-Raphson is used, which compares with Fisher scoring with expexp.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Warning

The standard errors produced by a summary of the model may be wrong.

Note

This family function works only for intercept-only models, i.e., y ~ 1 where y is the response.

The estimate of α is attached to the misc slot of the object, which is a list and contains the component shape.

As Newton-Raphson is used, the working weights are sometimes negative, and some adjustment is made to these to make them positive.

Like expexp, good initial values are needed. Convergence may be slow.

Author(s)

T. W. Yee

References

Gupta, R. D. and Kundu, D. (2001) Exponentiated exponential family: an alternative to gamma and Weibull distributions, *Biometrical Journal*, **43**, 117–130.

See Also

expexp.

Examples

```
# Ball bearings data (number of million revolutions before failure)
bbearings = c(17.88, 28.92, 33.00, 41.52, 42.12, 45.60,
48.80, 51.84, 51.96, 54.12, 55.56, 67.80, 68.64, 68.64,
68.88, 84.12, 93.12, 98.64, 105.12, 105.84, 127.92,
128.04, 173.40)
fit = vglm(bbearings ~ 1, expexp1(ishape=4), trace=TRUE,
           maxit=50, checkwz=FALSE)
coef(fit, matrix=TRUE)
Coef(fit) # Authors get c(0.0314, 5.2589) with log-lik -112.9763
fit@misc$shape    # Estimate of shape
```

```
logLik(fit)

# Failure times of the airconditioning system of an airplane
acplane = c(23, 261, 87, 7, 120, 14, 62, 47,
225, 71, 246, 21, 42, 20, 5, 12, 120, 11, 3, 14,
71, 11, 14, 11, 16, 90, 1, 16, 52, 95)
fit = vglm(acplane ~ 1, expexp1(ishape=0.8), trace=TRUE,
           maxit=50, checkwz=FALSE)
coef(fit, matrix=TRUE)
Coef(fit) # Authors get c(0.0145, 0.8130) with log-lik -152.264
fit@misc$shape    # Estimate of shape
logLik(fit)
```

`exponential` *Exponential Distribution*

Description

Maximum likelihood estimation for the exponential distribution.

Usage

```
exponential(link = "loge", earg = list(), location = 0, expected = TRUE)
```

Arguments

`link`	Parameter link function applied to the positive parameter $rate$. See `Links` for more choices.
`earg`	List. Extra argument for the link. See `earg` in `Links` for general information.
`location`	Numeric of length 1, the known location parameter, A, say.
`expected`	Logical. If `TRUE` Fisher scoring is used, otherwise Newton-Raphson. The latter is usually faster.

Details

The family function assumes the response Y has density

$$f(y) = \lambda \exp(-\lambda(y - A))$$

for $y > A$, where A is the known location parameter. By default, $A = 0$. Then $E(Y) = A + 1/\lambda$ and $Var(Y) = 1/\lambda^2$.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

Suppose $A = 0$. For a fixed time interval, the number of events is Poisson with mean λ if the time between events has a geometric distribution with mean λ^{-1}. The argument `rate` in `exponential` is the same as `rexp` etc. The argument `lambda` in `rpois` is somewhat the same as `rate` here.

Author(s)

T. W. Yee

References

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

See Also

`poissonff`, `freund61`.

Examples

```
nn = 100
x1 = runif(nn) - 0.5
x2 = runif(nn) - 0.5
eta = 0.2 - 0.7 * x1 + 1.9 * x2
rate = exp(eta)
y = rexp(nn, rate=rate)
stem(y)
fit = vglm(y ~ x1 + x2, exponential, trace=TRUE, crit="c") # slower
fit = vglm(y ~ x1 + x2, exponential(exp=FALSE), trace=TRUE, crit="c") # faster
coef(fit)
coef(fit, mat=TRUE)
Coef(fit)
summary(fit)
```

`fff` *F Distribution Family Function*

Description

Maximum likelihood estimation of the (2-parameter) F distribution.

Usage

```
fff(link="loge", earg=list(), idf1=NULL, idf2=NULL,
    method.init=1, zero=NULL)
```

Arguments

link	Parameter link function for both parameters. See Links for more choices. The default keeps the parameters positive.
earg	List. Extra argument for the link. See earg in Links for general information.
idf1, idf2	Numeric and positive. Initial value for the parameters. The default is to choose each value internally.
method.init	Initialization method. Either the value 1 or 2. If both fail try setting values for idf1 and idf2.
zero	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The value must be from the set {1,2}, corresponding respectively to $df1$ and $df2$. By default all linear/additive predictors are modelled as a linear combination of the explanatory variables.

Details

The F distribution is named after Fisher and has a density function that has two parameters, called df1 and df2 here. This function treats these degrees of freedom as *positive reals* rather than integers. The mean of the distribution is $df2/(df2-2)$ provided $df2 > 2$, and its variance is $2df2^2(df1+df2-2)/(df1(df2-2)^2(df2-4))$ provided $df2 > 4$. The estimated mean is returned as the fitted values. Although the F distribution can be defined to accommodate a non-centrality parameter ncp, it is assumed zero here.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Warning

Numerical problems will occur when the estimates of the parameters are too low.

Note

This family function uses the BFGS quasi-Newton update formula for the working weight matrices. Consequently the estimated variance-covariance matrix may be inaccurate or simply wrong! The standard errors must be therefore treated with caution; these are computed in functions such as vcov() and summary().

Author(s)

T. W. Yee

References

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

See Also

FDist.

Examples

```
x = runif(n <- 4000)
df1 = exp(2+0.5*x)
df2 = exp(2-0.5*x)
y = rf(n, df1, df2)
fit = vglm(y  ~ x, fff, trace=TRUE)
fit = vglm(y  ~ x, fff(link="logoff", earg=list(offset=0.5)), trace=TRUE)
coef(fit, matrix=TRUE)
Coef(fit)
vcov(fit)   # caution needed!
```

---

fgm *Farlie-Gumbel-Morgenstern's Bivariate Distribution Family Function*

Description

Estimate the association parameter of Farlie-Gumbel-Morgenstern's bivariate distribution using maximum likelihood estimation.

Usage

```
fgm(lapar="identity", earg=list(), iapar=NULL, method.init=1)
```

Arguments

lapar	Link function applied to the association parameter α, which is real. See Links for more choices.
earg	List. Extra argument for the link. See earg in Links for general information.
iapar	Numeric. Optional initial value for α. By default, an initial value is chosen internally. If a convergence failure occurs try assigning a different value. Assigning a value will override the argument method.init.
method.init	An integer with value 1 or 2 which specifies the initialization method. If failure to converge occurs try the other value, or else specify a value for ia.

Details

The cumulative distribution function is

$$P(Y_1 \leq y_1, Y_2 \leq y_2) = y_1 y_2 (1 + \alpha(1 - y_1)(1 - y_2))$$

for real α (the range is data-dependent). The support of the function is the unit square. The marginal distributions are the standard uniform distributions. When $\alpha = 0$ then the random variables are independent.

A variant of Newton-Raphson is used, which only seems to work for an intercept model. It is a very good idea to set trace=TRUE. This **VGAM** family function is prone to numerical difficulties.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

The response must be a two-column matrix. Currently, the fitted value is a matrix with two columns and values equal to 0.5. This is because each marginal distribution corresponds to a standard uniform distribution.

Author(s)

T. W. Yee

References

Castillo, E., Hadi, A. S., Balakrishnan, N. Sarabia, J. S. (2005) *Extreme Value and Related Models with Applications in Engineering and Science*, Hoboken, N.J.: Wiley-Interscience.

See Also

frank, morgenstern.

Examples

```
n = 1000
ymat = cbind(runif(n), runif(n))
## Not run: plot(ymat)
fit = vglm(ymat ~ 1, fam=fgm, trace=TRUE)
fit = vglm(ymat ~ 1, fam=fgm, trace=TRUE, crit="coef")
coef(fit, matrix=TRUE)
Coef(fit)
fitted(fit)[1:5,]
```

---

fill *Creates a Matrix of Appropriate Dimension*

Description

A support function for the argument xij, it generates a matrix of an appropriate dimension.

Usage

```
fill(x, values = 0, ncolx = ncol(x))
```

Arguments

x	A vector or matrix which is used to determine the dimension of the answer, in particular, the number of rows. After converting x to a matrix if necessary, the answer is a matrix of values values, of dimension nrow(x) by ncolx.
values	Numeric. The answer contains these values which are recycled if necessary.
ncolx	The number of columns of the returned matrix. The default is the number of columns of x.

Details

The xij argument for vglm allows the user to input variables specific to each linear predictor. For example, consider the bivariate logit model where the first/second linear/additive predictor is the logistic regression of the first/second binary response respectively. The third linear/additive predictor is log(OR) = eta3, where OR is the odds ratio. If one has ocular pressure as a covariate in this model then xij is required to handle the ocular pressure for each eye, since these will be different in general. [This contrasts with a variable such as age, the age of the person, which has a common value for both eyes.] In order to input these data into vglm one often finds that functions fill, fill1, etc. are useful.

All terms in the xij argument must appear in the main formula argument in vglm.

Value

matrix(values, nrow=nrow(x), ncol=ncolx), i.e., a matrix consisting of values values, with the number of rows matching x, and the default number of columns is the number of columns of x.

Warning

The use of the xij argument overrides other arguments such as exchangeable and zero. Care is needed in such cases. See the examples below.

Note

Additionally, there are currently 3 other identical fill functions, called fill1, fill2 and fill3; if you need more then assign fill4 = fill5 = fill1 etc. The reason for this is that if more than one fill function is needed then they must be unique. For example, if $M = 4$ then xij = op ~ lop + rop + fill(mop) + fill(mop) would reduce to xij = op ~ lop + rop + fill(mop), whereas xij = op ~ lop + rop + fill1(mop) + fill2(mop) would retain M terms, which is needed.

The constraint matrices, as returned by constraints, have a different meaning when xij is used.

In Examples 1 to 3 below, the xij argument illustrates covariates that are specific to a linear predictor. Here, lop/rop are the ocular pressures of the left/right eye in an artificial dataset, and mop is their mean. Variables leye and reye might be the presence/absence of a particular disease on the LHS/RHS eye respectively. Examples 1 and 2 are deliberately misspecified. The output from, e.g., coef(fit, matrix=TRUE), looks wrong but is correct because the coefficients are multiplied by the zeros produced from fill.

In Example 4, the `xij` argument illustrates fitting the model where there is a common smooth function of the ocular pressure. One should use regression splines since `s` in `vgam` does not handle the `xij` argument. However, regression splines such as `bs` and `ns` need to have the same knots here for both functions, and Example 4 illustrates a trick involving a function `BS` to obtain this. Although regression splines create more than a single column per term in the model matrix, `fill(BS(lop,rop,mop))` creates the required (same) number of columns.

Author(s)

T. W. Yee

References

More information can be found at http://www.stat.auckland.ac.nz/~yee.

See Also

`vglm`, `vglm.control`.

Examples

```
fill(runif(5))
fill(runif(5), ncol=3)
fill(runif(5), val=1, ncol=3)

# Generate eyes data for the examples below. Eyes are independent (OR=1).
set.seed(123)
n = 2000  # Number of people
eyes = data.frame(lop = round(runif(n), 2),
                  rop = round(runif(n), 2),
                  age = round(rnorm(n, 40, 10)))
eyes = transform(eyes,
    mop = (lop + rop) / 2, # mean ocular pressure
    eta1 = 0 - 2*lop + 0.04*age, # Linear predictor for left eye
    eta2 = 0 - 2*rop + 0.04*age) # Linear predictor for right eye
eyes = transform(eyes,
    leye = rbinom(n, size=1, prob=exp(eta1)/(1+exp(eta1))),
    reye = rbinom(n, size=1, prob=exp(eta2)/(1+exp(eta2))))

# Example 1
# Non-exchangeable errors (misspecified model)
fit1 = vglm(cbind(leye,reye) ~ lop + rop + fill(lop) + age,
            family = binom2.or(exchangeable=FALSE, zero=NULL),
            xij = op ~ lop + rop + fill(lop), data=eyes)
model.matrix(fit1, type="lm")[1:7,]   # LM model matrix
model.matrix(fit1, type="vlm")[1:7,]  # Big VLM model matrix
coef(fit1)
coef(fit1, matrix=TRUE)  # Looks wrong but is correct
coef(fit1, matrix=TRUE, compress=FALSE)  # Looks wrong but is correct
constraints(fit1)
max(abs(predict(fit1)-predict(fit1, new=eyes))) # Predicts correctly
summary(fit1)
```

```
# Example 2
# Nonexchangeable errors (misspecified model), OR is a function of mop
fit2 = vglm(cbind(leye,reye) ~ lop + rop + mop + age,
            family = binom2.or(exchangeable=FALSE, zero=NULL),
            xij = op ~ lop + rop + mop, data=eyes)
model.matrix(fit2, type="lm")[1:7,]   # LM model matrix
model.matrix(fit2, type="vlm")[1:7,]  # Big VLM model matrix
coef(fit2)
coef(fit2, matrix=TRUE)  # correct
coef(fit2, matrix=TRUE, compress=FALSE)  # correct
max(abs(predict(fit2)-predict(fit2, new=eyes))) # Predicts correctly
summary(fit2)

# Example 3. This model is correctly specified.
# Exchangeable errors
fit3 = vglm(cbind(leye,reye) ~ lop + rop + fill(lop) + age,
            family = binom2.or(exchangeable=TRUE, zero=3),
            xij = op ~ lop + rop + fill(lop), data=eyes)
model.matrix(fit3, type="lm")[1:7,]   # LM model matrix
model.matrix(fit3, type="vlm")[1:7,]  # Big VLM model matrix
coef(fit3)
coef(fit3, matrix=TRUE) # Looks wrong but is correct
coef(fit3, matrix=TRUE, compress=FALSE) # Looks wrong but is correct
predict(fit3, new=eyes[1:4,])  # Note the 'scalar' OR, i.e., zero=3
max(abs(predict(fit3)-predict(fit3, new=eyes))) # Predicts correctly
summary(fit3)

# Example 4. This model uses regression splines on ocular pressure.
# It assumes exchangeable errors.
BS = function(x, ...) bs(c(x,...), df=3)[1:length(x),]
fit4 = vglm(cbind(leye,reye) ~ BS(lop,rop,mop) + BS(rop,lop,mop) +
            fill(BS(lop,rop,mop)) + age,
            family = binom2.or(exchangeable=TRUE, zero=3),
            xij = BS(op) ~ BS(lop,rop,mop) + BS(rop,lop,mop) +
                  fill(BS(lop,rop,mop)), data=eyes)
model.matrix(fit4, type="lm")[1:7,]   # LM model matrix
model.matrix(fit4, type="vlm")[1:7,]  # Big VLM model matrix
coef(fit4)
coef(fit4, matrix=TRUE) # Looks wrong but is correct
coef(fit4, matrix=TRUE, compress=FALSE) # Looks wrong but is correct
predict(fit4, new=eyes[1:4,])  # Note the 'scalar' OR, i.e., zero=3
max(abs(predict(fit4)-predict(fit4, new=eyes))) # Predicts correctly
summary(fit4)
```

`fisherz` *Fisher's Z Link Function*

Description

Computes the Fisher Z transformation, including its inverse and the first two derivatives.

Usage

```
fisherz(theta, earg = list(), inverse = FALSE, deriv = 0,
        short = TRUE, tag = FALSE)
```

Arguments

theta	Numeric or character. See below for further details.
earg	Optional list. Extra argument for passing in additional information. Values of theta which are less than or equal to -1 can be replaced by the bminvalue component of the list earg before computing the link function value. Values of theta which are greater than or equal to 1 can be replaced by the bmaxvalue component of the list earg before computing the link function value. See Links for general information about earg.
inverse	Logical. If TRUE the inverse function is computed.
deriv	Order of the derivative. Integer with value 0, 1 or 2.
short	Used for labelling the blurb slot of a vglmff-class object.
tag	Used for labelling the linear/additive predictor in the initialize slot of a vglmff-class object. Contains a little more information if TRUE.

Details

The fisherz link function is commonly used for parameters that lie between -1 and 1. Numerical values of theta close to -1 or 1 or out of range result in Inf, -Inf, NA or NaN. The arguments short and tag are used only if theta is character.

Value

For deriv = 0, 0.5 * log((1+theta)/(1-theta)) when inverse = FALSE, and if inverse = TRUE then (exp(2*theta)-1)/(exp(2*theta)+1).

For deriv = 1, then the function returns *d* theta / *d* eta as a function of theta if inverse = FALSE, else if inverse = TRUE then it returns the reciprocal.

Here, all logarithms are natural logarithms, i.e., to base *e*.

Note

Numerical instability may occur when theta is close to -1 or 1. One way of overcoming this is to use earg.

The link function rhobit is very similar to fisherz, e.g., just twice the value of fisherz.

Author(s)

Thomas W. Yee

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

See Also

Links, rhobit, logit.

Examples

```
theta = seq(-0.99, 0.99, by=0.01)
y = fisherz(theta)
## Not run:
plot(theta, y, type="l", las=1, ylab="", main="fisherz(theta)")
abline(v=0, h=0, lty=2)
## End(Not run)

x = c(seq(-1.02, -0.98, by=0.01), seq(0.97, 1.02, by=0.01))
fisherz(x)  # Has NAs
fisherz(x, earg=list(bminvalue= -1 + .Machine$double.eps,
                     bmaxvalue=  1 - .Machine$double.eps))  # Has no NAs
```

`fisk` *Fisk Distribution family function*

Description

Maximum likelihood estimation of the 2-parameter Fisk distribution.

Usage

```
fisk(link.a = "loge", link.scale = "loge",
     earg.a=list(), earg.scale=list(),
     init.a = NULL, init.scale = NULL, zero = NULL)
```

Arguments

`link.a, link.scale`
: Parameter link functions applied to the (positive) parameters `a` and `scale`. See Links for more choices.

`earg.a, earg.scale`
: List. Extra argument for each of the links. See `earg` in Links for general information.

`init.a, init.scale`
: Optional initial values for `a` and `scale`.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. Here, the values must be from the set {1,2} which correspond to `a`, `scale`, respectively.

Details

The 2-parameter Fisk (aka log-logistic) distribution is the 4-parameter generalized beta II distribution with shape parameter $q = p = 1$. It is also the 3-parameter Singh-Maddala distribution with shape parameter $q = 1$, as well as the Dagum distribution with $p = 1$. More details can be found in Kleiber and Kotz (2003).

The Fisk distribution has density

$$f(y) = ay^{a-1}/[b^a\{1 + (y/b)^a\}^2]$$

for $a > 0$, $b > 0$, $y > 0$. Here, b is the scale parameter `scale`, and `a` is a shape parameter. The cumulative distribution function is

$$F(y) = 1 - [1 + (y/b)^a]^{-1} = [1 + (y/b)^{-a}]^{-1}.$$

The mean is

$$E(Y) = b\,\Gamma(1 + 1/a)\,\Gamma(1 - 1/a)$$

provided $a > 1$.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

If the self-starting initial values fail, try experimenting with the initial value arguments, especially those whose default value is not `NULL`.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

See Also

`Fisk`, `genbetaII`, `betaII`, `dagum`, `sinmad`, `invlomax`, `lomax`, `paralogistic`, `invparalogistic`.

Examples

```
y = rfisk(n=200, 4, 6)
fit = vglm(y ~ 1, fisk, trace=TRUE)
fit = vglm(y ~ 1, fisk(init.a=3.3), trace=TRUE, crit="c")
coef(fit, mat=TRUE)
Coef(fit)
summary(fit)
```

`fitted.vlm` *Fitted Values of a VLM object*

Description

Extractor function for the fitted values of a model object that inherits from a *vector linear model* (VLM), e.g., a model of class `"vglm"`.

Usage

```
fitted.vlm(object, matrix = TRUE, ...)
```

Arguments

`object`	a model object that inherits from a VLM.
`matrix`	Logical. Return the answer as a matrix? If `FALSE` then it will be a vector.
`...`	Currently unused.

Details

The "fitted values" usually corresponds to the mean response, however, because the **VGAM** package fits so many models, this sometimes refers to quantities such as quantiles. It may even not exist, e.g., for a Cauchy distribution.

Value

The fitted values as returned by the `inverse` slot of the **VGAM** family function, evaluated at the final IRLS iteration.

Note

This function is one of several extractor functions for the **VGAM** package. Others include `coef`, `deviance`, `weights` and `constraints` etc. This function is equivalent to the methods function for the generic function `fitted.values`.

If `fit` is a VLM or VGLM then `fitted(fit)` and `predict(fit, type="response")` should be equivalent. The latter has the advantage in that it handles a `newdata` argument so that the fitted values can be computed for a different data set.

Author(s)

Thomas W. Yee

References

Chambers, J. M. and T. J. Hastie (eds) (1992) *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

fitted, predict.vglm, vglmff-class.

Examples

```
# Categorical regression example 1
data(pneumo)
pneumo = transform(pneumo, let=log(exposure.time))
fit = vglm(cbind(normal, mild, severe) ~ let,
           cumulative(parallel=TRUE, reverse=TRUE), pneumo)
fit
fitted(fit)

# LMS quantile regression example 2
data(bminz)
fit = vgam(BMI ~ s(age, df=c(4,2)),
           fam=lms.bcn(zero=1), data=bminz, trace=TRUE)
# The following are equal
predict(fit, type="r")[1:3,]
fitted(fit)[1:3,]
predict(fit, type="r", newdata=bminz[1:3,])
```

---

frank *Frank's Bivariate Distribution Family Function*

Description

Estimate the association parameter of Frank's bivariate distribution using maximum likelihood estimation.

Usage

```
frank(lapar="loge", eapar=list(), iapar=2)
```

Arguments

lapar	Link function applied to the (positive) association parameter α. See Links for more choices.
eapar	List. Extra argument for the link. See earg in Links for general information.
iapar	Numeric. Initial value for α. If a convergence failure occurs try assigning a different value.

Details

The cumulative distribution function is

$$P(Y_1 \leq y_1, Y_2 \leq y_2) = H_\alpha(y_1, y_2) = \log_\alpha[1 + (\alpha^{y_1} - 1)(\alpha^{y_2} - 1)/(\alpha - 1)]$$

for $\alpha \neq 1$. Note the logarithm here is to base α. The support of the function is the unit square.

When $0 < \alpha < 1$ the probability density function $h_\alpha(y_1, y_2)$ is symmetric with respect to the lines $y_2 = y_1$ and $y_2 = 1 - y_1$. When $\alpha > 1$ then $h_\alpha(y_1, y_2) = h_{1/\alpha}(1 - y_1, y_2)$.

If $\alpha = 1$ then $H(y_1, y_2) = y_1 y_2$, i.e., uniform on the unit square. As α approaches 0 then $H(y_1, y_2) = \min(y_1, y_2)$. As α approaches infinity then $H(y_1, y_2) = \max(0, y_1 + y_2 - 1)$.

A variant of Newton-Raphson is used, which only seems to work for an intercept model.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Note

The response must be a two-column matrix. Currently, the fitted value is a matrix with two columns and values equal to a half. This is because the marginal distributions correspond to a standard uniform distribution.

Author(s)

T. W. Yee

References

Genest, C. (1987) Frank's family of bivariate distributions. *Biometrika*, **74**, 549–555.

See Also

`rfrank`, `fgm`.

Examples

```
ymat = rfrank(n=2000, alpha=exp(4))
## Not run: plot(ymat)
fit = vglm(ymat ~ 1, fam=frank, trace=TRUE)
coef(fit, matrix=TRUE)
Coef(fit)
vcov(fit)
fitted(fit)[1:5,]
summary(fit)
```

Frank *Frank's Bivariate Distribution*

Description

Density, distribution function, quantile function and random generation for the one parameter Frank distribution.

Usage

```
dfrank(x1, x2, alpha)
pfrank(q1, q2, alpha)
rfrank(n, alpha)
```

Arguments

`x1, x2, q1, q2`	vector of quantiles.
`n`	number of observations. Must be a positive integer of length 1.
`alpha`	the positive association parameter α.

Details

See `frank`, the **VGAM** family functions for estimating the association parameter by maximum likelihood estimation, for the formula of the cumulative distribution function and other details.

Value

`dfrank` gives the density, `pfrank` gives the distribution function, and `rfrank` generates random deviates (a two-column matrix).

Author(s)

T. W. Yee

References

Genest, C. (1987) Frank's family of bivariate distributions. *Biometrika*, **74**, 549–555.

See Also

`frank`.

Examples

```
## Not run:
N = 100
x = seq(-0.30, 1.30, len=N)
alpha = 8
ox = expand.grid(x, x)
z = dfrank(ox[,1], ox[,2], alp=alpha)
contour(x, x, matrix(z, N, N))
z = pfrank(ox[,1], ox[,2], alp=alpha)
contour(x, x, matrix(z, N, N))

alpha = exp(4)
plot(r <- rfrank(n=3000, alpha=alpha))
par(mfrow=c(1,2))
hist(r[,1]) # Should be uniform
hist(r[,2]) # Should be uniform
## End(Not run)
```

---

frechet *Frechet Distribution Family Function*

Description

Maximum likelihood estimation of the 2-parameter and 3-parameter Frechet distribution.

Usage

```
frechet2(location=0, lscale="loge", lshape="loglog",
         escale = list(), eshape = list(),
         iscale=NULL, ishape=3, zero=NULL)
frechet3(anchor=NULL, ldifference="loge",
         lscale="loge", lshape="loglog",
         edifference=list(), escale = list(), eshape = list(),
         ilocation=NULL, iscale=NULL, ishape=3, zero=NULL,
         effpos = .Machine$double.eps^0.75)
```

Arguments

`location` Numeric. Location parameter. It is called a below.

`lscale, lshape`
Link functions for the parameters. See `Links` for more choices.

`iscale, ishape`
Initial value for the scale and shape parameters respectively. Both parameters must be positive. A `NULL` means it is chosen internally.

`edifference, escale, eshape`
Extra argument for the respective links. See `earg` in `Links` for general information.

`zero` An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The value must be from the set {1,2}, corresponding respectively to $scale$ and $shape$. By default all linear/additive predictors are modelled as a linear combination of the explanatory variables.

`anchor` An "anchor" point for estimating the location parameter. This must be a value no greater than `min(y)` where `y` is the response. The location parameter is $A - D$ where A is the anchor, D is the "difference" (default is to make this positive). The default value of `anchor` means `min(y)` is chosen.

`ldifference` Parameter link function for the difference D between the anchor point and the location parameter estimate. The default keeps this difference positive so that numerical problems are less likely to occur.

`ilocation` Optional initial value for the location parameter. A good choice can speed up the convergence rate markedly. A `NULL` means it is chosen internally.

`effpos` A small number, usually positive, to pass into the quasi-Newton updating function. A value greater than `effpos` is considered effectively positive.

Details

The Frechet distribution has a density function that can be written

$$f(y) = \frac{sb}{(y-a)^2}[b/(y-a)]^{s-1} \exp[-(b/(y-a))^s]$$

for $y > a$ and scale parameter $b > 0$. The positive shape parameter is s. The cumulative distribution function is

$$F(y) = \exp[-(b/(y-a))^s].$$

The mean of Y is $a + b\Gamma(1 - 1/s)$ for $s > 1$ (these are returned as the fitted values). Note that the `loglog` link ensures $s > 1$. The variance of Y is $b^2[\Gamma(1 - 2/s) - \Gamma^2(1 - 1/s)]$ for $s > 2$.

`frechet2` has a known whereas `frechet3` estimates it. Estimating a well requires a lot of data and a good choice of `ilocation` will help speed up convergence. For `frechet3` the default linear/additive predictors are $\log(D)$, $\log(b)$, $\log\log(s)$, respectively. It would be great if the first linear/additive predictor was a direct function of the location parameter, but this can run the risk that the estimate is out of range (i.e., greater than `min(y)`).

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Warning

Convergence for `frechet3` can be very slow, especially if the initial value for the location parameter is poor. Setting something like `maxit=200, trace=TRUE` is a good idea.

Note

These family functions use the BFGS quasi-Newton update formula for the working weight matrices. Consequently the estimated variance-covariance matrix may be inaccurate or simply wrong! The standard errors must be therefore treated with caution; these are computed in functions such as `vcov()` and `summary()`.

If `fit` is a `frechet3` fit then `fit@extra$location` is the final estimate of the location parameter, and `fit@extra$LHSanchor` is the anchor point.

Author(s)

T. W. Yee

References

Castillo, E., Hadi, A. S., Balakrishnan, N. Sarabia, J. S. (2005) *Extreme Value and Related Models with Applications in Engineering and Science*, Hoboken, N.J.: Wiley-Interscience.

See Also

`rfrechet`.

Examples

```
y = rfrechet(n <- 1000, shape=exp(exp(0)))
## Not run: hist(y)
fit2 = vglm(y ~ 1, frechet2, trace=TRUE, maxit=155)
fit3 = vglm(y ~ 1, frechet3(ilocation=0), trace=TRUE, maxit=155)
coef(fit3, matrix=TRUE)
Coef(fit3)
fitted(fit3)[1:5,]
mean(y)
weights(fit3, type="w")[1:5,]
vcov(fit3)   # caution needed!
fit3@extra$location[1:3]  # Estimate of the location parameter
fit3@extra$LHSanchor  # Anchor point
min(y)
```

Frechet *The Frechet Distribution*

Description

Density, distribution function, quantile function and random generation for the three parameter Frechet distribution.

Usage

```
dfrechet(x, location=0, scale=1, shape)
pfrechet(q, location=0, scale=1, shape)
qfrechet(p, location=0, scale=1, shape)
rfrechet(n, location=0, scale=1, shape)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. Must be a positive integer of length 1.
`location, scale, shape`	the location parameter a, scale parameter b, and shape parameter s.

Details

See `frechet2` and `frechet3`, the **VGAM** family functions for estimating the 2 (without location parameter) and 3 parameters by maximum likelihood estimation, for the formula of the probability density function and range restrictions on the parameters.

Value

`dfrechet` gives the density, `pfrechet` gives the distribution function, `qfrechet` gives the quantile function, and `rfrechet` generates random deviates.

Author(s)

T. W. Yee

References

Castillo, E., Hadi, A. S., Balakrishnan, N. Sarabia, J. S. (2005) *Extreme Value and Related Models with Applications in Engineering and Science*, Hoboken, N.J.: Wiley-Interscience.

See Also

`frechet2`, `frechet3`.

Examples

```
## Not run:
shape = 5
x = seq(-0.1, 3.5, len=100)
plot(x, dfrechet(x, shape=shape), type="l", ylab="", las=1,
     main="Frechet density divided into 10 equal areas; red=cdf")
abline(h=0, col="blue", lty=2)
qq = qfrechet(seq(0.1,0.9,by=0.1), shape=shape)
lines(qq, dfrechet(qq, shape=shape), col="purple", lty=3, type="h")
lines(x, pfrechet(q=x, shape=shape), col="red")
## End(Not run)
```

freund61 *Freund's (1961) Bivariate Extension of the Exponential Distribution*

Description

Estimate the four parameters of the Freund (1961) bivariate extension of the exponential distribution using maximum likelihood estimation.

Usage

```
freund61(la="loge", lap="loge", lb="loge", lbp="loge",
         ia=NULL, iap=NULL, ib=NULL, ibp=NULL,
         independent=FALSE, zero=NULL)
```

Arguments

`la,lap,lb,lbp`
: Link function applied to the (positive) parameters α, α', β and β', respectively (the "p" stands for "prime"). See `Links` for more choices.

`ia,iap,ib,ibp`
: Initial value for the four parameters respectively. The default is to estimate them all internally.

`independent`
: Logical. If `TRUE` then the parameters are constrained to satisfy $\alpha = \alpha'$ and $\beta = \beta'$, which implies that y_1 and y_2 are independent and each have an ordinary exponential distribution.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,3,4}. The default is none of them.

Details

This model represents one type of bivariate extension of the exponential distribution that is applicable to certain problems, in particular, to two-component systems which can function if one of the components has failed. For example, engine failures in two-engine planes, paired organs such as peoples' eyes, ears and kidneys. Suppose y_1 and y_2 are random variables representing the lifetimes of two components A and B in a two component system. The dependence between y_1 and y_2 is essentially such that the failure of the B component changes the parameter of the exponential life distribution of the A component from α to α', while the failure of the A component changes the parameter of the exponential life distribution of the B component from β to β'.

The joint probability density function is given by

$$f(y_1, y_2) = \alpha\beta' \exp(-\beta' y_2 - (\alpha + \beta - \beta')y_1)$$

for $0 < y_1 < y_2$, and

$$f(y_1, y_2) = \beta\alpha' \exp(-\alpha' y_1 - (\alpha + \beta - \alpha')y_2)$$

for $0 < y_2 < y_1$. Here, all four parameters are positive, as well as the responses y_1 and y_2. Under this model, the probability that component A is the first to fail is $\alpha/(\alpha+\beta)$. The time to the first failure is distributed as an exponential distribution with rate $\alpha+\beta$. Furthermore, the distribution of the time from first failure to failure of the other component is a mixture of Exponential(α') and Exponential(β') with proportions $\beta/(\alpha+\beta)$ and $\alpha/(\alpha+\beta)$ respectively.

The marginal distributions are, in general, not exponential. By default, the linear/additive predictors are $\eta_1 = \log(\alpha)$, $\eta_2 = \log(\alpha')$, $\eta_3 = \log(\beta)$, $\eta_4 = \log(\beta')$.

A special case is when $\alpha = \alpha'$ and $\beta = \beta'$, which means that y_1 and y_2 are independent, and both have an ordinary exponential distribution with means $1/\alpha$ and $1/\beta$ respectively.

Fisher scoring is used, and the initial values correspond to the MLEs of an intercept model. Consequently, convergence may take only one iteration.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Note

To estimate all four parameters, it is necessary to have some data where $y_1 < y_2$ and $y_2 < y_1$.

The response must be a two-column matrix, with columns y_1 and y_2. Currently, the fitted value is a matrix with two columns; the first column has values $(\alpha'+\beta)/(\alpha'(\alpha+\beta))$ for the mean of y_1, while the second column has values $(\beta'+\alpha)/(\beta'(\alpha+\beta))$ for the mean of y_2. The variance of y_1 is

$$\frac{(\alpha')^2 + 2\alpha\beta + \beta^2}{(\alpha')^2(\alpha+\beta)^2},$$

the variance of y_2 is

$$\frac{(\beta')^2 + 2\alpha\beta + \alpha^2}{(\beta')^2(\alpha+\beta)^2},$$

the covariance of y_1 and y_2 is

$$\frac{\alpha'\beta' - \alpha\beta}{\alpha'\beta'(\alpha+\beta)^2}.$$

Author(s)

T. W. Yee

References

Freund, J. E. (1961) A bivariate extension of the exponential distribution. *Journal of the American Statistical Association*, **56**, 971–977.

See Also

`exponential`.

Examples

```
y1 = rexp(n <- 200, rate=4)
y2 = rexp(n, rate=8)
ymat = cbind(y1,y2)
fit =  vglm(ymat ~ 1, fam=freund61, trace=TRUE)
coef(fit, matrix=TRUE)
Coef(fit)
vcov(fit)
fitted(fit)[1:5,]
summary(fit)

# y1 and y2 are independent, so fit an independence model
fit2 = vglm(ymat ~ 1, fam=freund61(indep=TRUE), trace=TRUE)
coef(fit2, matrix=TRUE)
constraints(fit2)
1 - pchisq(2*(logLik(fit)-logLik(fit2)), df=2) # p-value
```

---

fsqrt *Folded Square Root Link Function*

Description

Computes the folded square root transformation, including its inverse and the first two derivatives.

Usage

```
fsqrt(theta, earg = list(min=0, max=1, mux=sqrt(2)),
      inverse = FALSE, deriv = 0, short = TRUE, tag = FALSE)
```

Arguments

theta	Numeric or character. See below for further details.
earg	List with components min, max and mux. These are called L, U and K below.
inverse	Logical. If TRUE the inverse function is computed.
deriv	Order of the derivative. Integer with value 0, 1 or 2.
short	Used for labelling the blurb slot of a vglmff-class object.
tag	Used for labelling the linear/additive predictor in the initialize slot of a vglmff-class object. Contains a little more information if TRUE.

Details

The folded square root link function can be applied to parameters that lie between L and U inclusive. Numerical values of theta out of range result in NA or NaN.

The arguments short and tag are used only if theta is character.

Value

For `fsqrt` with `deriv = 0`: $K(\sqrt{\theta - L} - \sqrt{U - \theta})$ or `mux * (sqrt(theta-min) - sqrt(max-theta))` when `inverse = FALSE`, and if `inverse = TRUE` then some more complicated function that returns a `NA` unless `theta` is between `-mux*sqrt(max-min)` and `mux*sqrt(max-min)`.

For `deriv = 1`, then the function returns *d* `theta` / *d* `eta` as a function of `theta` if `inverse = FALSE`, else if `inverse = TRUE` then it returns the reciprocal.

Note

The default has, if `theta` is 0 or 1, the link function value is `-sqrt(2)` and `+sqrt(2)` respectively. These are finite values, therefore one cannot use this link function for general modelling of probabilities because of numerical problem, e.g., with `binomialff`, `cumulative`. See the example below.

Author(s)

Thomas W. Yee

See Also

`Links`.

Examples

```
p = seq(0.01, 0.99, by=0.01)
fsqrt(p)
max(abs(fsqrt(fsqrt(p), inverse=TRUE) - p)) # Should be 0

p = c(seq(-0.02, 0.02, by=0.01), seq(0.97, 1.02, by=0.01))
fsqrt(p)  # Has NAs

## Not run:
p = seq(0.01, 0.99, by=0.01)
par(mfrow=c(2,2))
y = seq(-4, 4, length=100)
for(d in 0:1) {
    matplot(p, cbind(logit(p, deriv=d), fsqrt(p, deriv=d)),
            type="n", col="purple", ylab="transformation",
            lwd=2, las=1,
            main=if(d==0) "Some probability link functions"
            else "First derivative")
    lines(p, logit(p, deriv=d), col="limegreen", lwd=2)
    lines(p, probit(p, deriv=d), col="purple", lwd=2)
    lines(p, cloglog(p, deriv=d), col="chocolate", lwd=2)
    lines(p, fsqrt(p, deriv=d), col="tan", lwd=2)
    if(d==0) {
        abline(v=0.5, h=0, lty="dashed")
        legend(0, 4.5, c("logit", "probit", "cloglog", "fsqrt"),
               col=c("limegreen","purple","chocolate", "tan"), lwd=2)
    } else
```

```
        abline(v=0.5, lty="dashed")
}

for(d in 0) {
    matplot(y, cbind(logit(y, deriv=d, inverse=TRUE),
                     fsqrt(y, deriv=d, inverse=TRUE)),
            type="n", col="purple", xlab="transformation", ylab="p",
            lwd=2, las=1,
            main=if(d==0) "Some inverse probability link functions"
            else "First derivative")
    lines(y, logit(y, deriv=d, inverse=TRUE), col="limegreen", lwd=2)
    lines(y, probit(y, deriv=d, inverse=TRUE), col="purple", lwd=2)
    lines(y, cloglog(y, deriv=d, inverse=TRUE), col="chocolate", lwd=2)
    lines(y, fsqrt(y, deriv=d, inverse=TRUE), col="tan", lwd=2)
    if(d==0) {
        abline(h=0.5, v=0, lty="dashed")
        legend(-4, 1, c("logit", "probit", "cloglog", "fsqrt"),
               col=c("limegreen","purple","chocolate", "tan"), lwd=2)
    }
}
## End(Not run)

# This is lucky to converge
earg = list(min=0, max=1, mux=5)
data(hunua)
fit.h = vglm(agaaus ~ bs(altitude),
             fam= binomialff(link="fsqrt", earg=earg),
             data=hunua, trace=TRUE, crit="d")
## Not run:
plotvgam(fit.h, se=TRUE, lcol="red", scol="red",
     main="Red is Hunua, Blue is Waitakere")
## End(Not run)
predict(fit.h, hunua, type="response")[1:3]

## Not run:
# The following fails.
data(pneumo)
pneumo = transform(pneumo, let=log(exposure.time))
earg = list(min=0, max=1, mux=10)
fit = vglm(cbind(normal, mild, severe) ~ let,
           cumulative(link="fsqrt", earg=earg, par=TRUE, rev=TRUE),
           data = pneumo, trace=TRUE, maxit=200)
## End(Not run)
```

`gamma1` *1-parameter Gamma Distribution*

Description

Estimates the 1-parameter gamma distribution by maximum likelihood estimation.

Usage

```
gamma1(link = "loge", earg=list())
```

Arguments

link	Link function applied to the (positive) *shape* parameter. See Links for more choices.
earg	List. Extra argument for the link. See earg in Links for general information.

Details

The density function is given by

$$f(y) = \exp(-y) \times y^{shape-1}/\Gamma(shape)$$

for $shape > 0$ and $y > 0$. Here, $\Gamma(shape)$ is the gamma function, as in gamma. The mean of Y (returned as the fitted values) is $\mu = shape$, and the variance is $\sigma^2 = shape$.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

This **VGAM** family function can handle a multivariate (matrix) response.

The parameter $shape$ matches with shape in rgamma. The argument rate in rgamma is assumed 1 for this family function.

If $rate$ is unknown use the family function gamma2.ab to estimate it too.

Author(s)

T. W. Yee

References

Most standard texts on statistical distributions describe the 1-parameter gamma distribution, e.g.,

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

See Also

gamma2.ab for the 2-parameter gamma distribution.

Examples

```
y = rgamma(n=100, shape= exp(3))
fit = vglm(y ~ 1, gamma1, trace=TRUE, crit="c")
coef(fit, matrix=TRUE)
Coef(fit)
summary(fit)
```

---

gamma2 *2-parameter Gamma Distribution*

Description

Estimates the 2-parameter gamma distribution by maximum likelihood estimation.

Usage

```
gamma2(lmu = "loge", lshape = "loge",
       emu = list(), eshape = list(),
       method.init = 1, deviance.arg = FALSE,
       ishape = NULL, zero = -2)
```

Arguments

`lmu, lshape`	Link functions applied to the (positive) *mu* and *shape* parameters (called μ and λ respectively). See `Links` for more choices.
`emu, eshape`	List. Extra argument for each of the links. See `earg` in `Links` for general information.
`ishape`	Optional initial value for *shape*. A `NULL` means a value is computed internally. If a failure to converge occurs, try using this argument. This argument is ignored if used within `cqo`; see the `iShape` argument of `qrrvglm.control` instead.
`method.init`	An integer with value `1` or `2` which specifies the initialization method for the μ parameter. If failure to converge occurs try another value (and/or specify a value for `ishape`).
`deviance.arg`	Logical. If `TRUE`, the deviance function is attached to the object. Under ordinary circumstances, it should be left alone because it really assumes the shape parameter is at the maximum likelihood estimate. Consequently, one cannot use that criterion to minimize within the IRLS algorithm. It should be set `TRUE` only when used with `cqo` under the fast algorithm.
`zero`	Integer valued vector, usually assigned -2 or 2 if used at all. Specifies which of the two linear/additive predictors are modelled as an intercept only. By default, the shape parameter (after `lshape` is applied) is modelled as a single unknown number that is estimated. It can be modelled as a function of the explanatory variables by setting `zero=NULL`. A negative value means that the value is recycled, so setting -2 means all shape parameters are intercept only.

Details

This distribution can model continuous skewed responses. The density function is given by

$$f(y;\mu,\lambda) = \frac{\exp(-\lambda y/\mu) \times (\lambda y/\mu)^{\lambda-1} \times \lambda}{\mu \times \Gamma(\lambda)}$$

for $\mu > 0$, $\lambda > 0$ and $y > 0$. Here, $\Gamma(\cdot)$ is the gamma function, as in `gamma`. The mean of Y is $\mu = \mu$ (returned as the fitted values) with variance $\sigma^2 = \mu^2/\lambda$. If $0 < \lambda < 1$ then the density has a pole at the origin and decreases monotonically as y increases. If $\lambda = 1$ then this corresponds to the exponential distribution. If $\lambda > 1$ then the density is zero at the origin and is unimodal with mode at $y = \mu - \mu/\lambda$; this can be achieved with `lshape="loglog"`.

By default, the two linear/additive predictors are $\eta_1 = \log(\mu)$ and $\eta_2 = \log(\lambda)$. This family function implements Fisher scoring and the working weight matrices are diagonal.

This **VGAM** family function handles *multivariate* responses, so that a matrix can be used as the response. The number of columns is the number of species, say, and `zero=-2` means that *all* species have a shape parameter equalling a (different) intercept only.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Note

The response must be strictly positive. A moment estimator for the shape parameter may be implemented in the future.

If `mu` and `shape` are vectors, then `rgamma(n=n, shape=shape, scale=mu/shape)` will generate random gamma variates of this parameterization, etc.; see `GammaDist`.

For `cqo` and `cao`, taking the logarithm of the response means (approximately) a `gaussianff` family may be used on the transformed data.

Author(s)

T. W. Yee

References

The parameterization of this **VGAM** family function is the 2-parameter gamma distribution described in the monograph

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

See Also

`gamma1` for the 1-parameter gamma distribution, `gamma2.ab` for another parameterization of the 2-parameter gamma distribution, `mckaygamma2` for *a* bivariate gamma distribution, `expexp`, `GammaDist`, `golf`.

Examples

```
# Essentially a 1-parameter gamma
y = rgamma(n <- 100, shape= exp(1))
fit1 = vglm(y ~ 1, gamma1, trace=TRUE, crit="c")
fit2 = vglm(y ~ 1, gamma2, trace=TRUE, crit="c")
coef(fit1, matrix=TRUE)
Coef(fit1)
coef(fit2, matrix=TRUE)
Coef(fit2)

# Essentially a 2-parameter gamma
y = rgamma(n <- 500, rate=exp(1), shape=exp(2))
fit2 = vglm(y ~ 1, gamma2, trace=TRUE, crit="c")
coef(fit2, matrix=TRUE)
Coef(fit2)
summary(fit2)
```

---

gamma2.ab *2-parameter Gamma Distribution*

Description

Estimates the 2-parameter gamma distribution by maximum likelihood estimation.

Usage

```
gamma2.ab(lrate = "loge", lshape = "loge",
          erate=list(), eshape=list(),
          irate=NULL, ishape=NULL, expected = TRUE, zero = 2)
```

Arguments

`lrate, lshape`
: Link functions applied to the (positive) *rate* and *shape* parameters. See `Links` for more choices.

`erate, eshape`
: List. Extra arguments for the links. See `earg` in `Links` for general information.

`expected`
: Logical. Use Fisher scoring? The default is yes, otherwise Newton-Raphson is used.

`irate, ishape`
: Optional initial values for *rate* and *shape*. A `NULL` means a value is computed internally. If a failure to converge occurs, try using these arguments.

`zero`
: An integer specifying which linear/additive predictor is to be modelled as an intercept only. If assigned, the single value should be either 1 or 2 or `NULL`. The default is to model $shape$ as an intercept only. A value `NULL` means neither 1 or 2.

Details

The density function is given by

$$f(y) = \exp(-rate \times y) \times y^{shape-1} \times rate^{shape}/\Gamma(shape)$$

for $shape > 0$, $rate > 0$ and $y > 0$. Here, $\Gamma(shape)$ is the gamma function, as in `gamma`. The mean of Y is $\mu = shape/rate$ (returned as the fitted values) with variance $\sigma^2 = \mu^2/shape = shape/rate^2$. By default, the two linear/additive predictors are $\eta_1 = \log(rate)$ and $\eta_2 = \log(shape)$.

The argument `expected` refers to the type of information matrix. The expected information matrix corresponds to Fisher scoring and is numerically better here. The observed information matrix corresponds to the Newton-Raphson algorithm and may be withdrawn from the family function in the future. If both algorithms work then the differences in the results are often not huge.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Note

The parameters $rate$ and $shape$ match with the arguments `rate` and `shape` of `rgamma`. Often, $scale = 1/rate$ is used.

If $rate = 1$ use the family function `gamma1` to estimate $shape$.

Author(s)

T. W. Yee

References

Most standard texts on statistical distributions describe the 2-parameter gamma distribution, e.g.,

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

See Also

`gamma1` for the 1-parameter gamma distribution, `gamma2` for another parameterization of the 2-parameter gamma distribution, `mckaygamma2` for *a* bivariate gamma distribution, `expexp`.

Examples

```
# Essentially a 1-parameter gamma
y = rgamma(n <- 100, shape= exp(1))
fit1 = vglm(y ~ 1, gamma1, trace=TRUE, crit="c")
fit2 = vglm(y ~ 1, gamma2.ab, trace=TRUE, crit="c")
coef(fit1, matrix=TRUE)
Coef(fit1)
coef(fit2, matrix=TRUE)
Coef(fit2)
```

```
# Essentially a 2-parameter gamma
y = rgamma(n <- 500, rate=exp(1), shape=exp(2))
fit2 = vglm(y ~ 1, gamma2.ab, trace=TRUE, crit="c")
coef(fit2, matrix=TRUE)
Coef(fit2)
summary(fit2)
```

`gammahyp` *Gamma Hyperbola Bivariate Distribution*

Description

Estimate the parameter of a gamma hyperbola bivariate distribution using maximum likelihood estimation.

Usage

```
gammahyp(ltheta="loge", itheta=NULL, expected=FALSE)
```

Arguments

`ltheta`	Link function applied to the (positive) parameter θ. See `Links` for more choices.
`itheta`	Initial value for the parameter. The default is to estimate it internally.
`expected`	Logical. `FALSE` means the Newton-Raphson (using the observed information matrix) algorithm, otherwise the expected information matrix is used (Fisher scoring algorithm).

Details

The joint probability density function is given by

$$f(y_1, y_2) = \exp(-e^{-\theta} y_1/\theta - \theta y_2)$$

for $\theta > 0$, $y_1 > 0$, $y_2 > 1$. The random variables Y_1 and Y_2 are independent. The marginal distribution of Y_1 is an exponential distribution with rate parameter $\exp(-\theta)/\theta$. The marginal distribution of Y_2 is an exponential distribution that has been shifted to the right by 1 and with rate parameter θ. The fitted values are stored in a two-column matrix with the marginal means, which are $\theta \exp(\theta)$ and $1 + 1/\theta$.

The default algorithm is Newton-Raphson because Fisher scoring tends to be much slower for this distribution.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Note

The response must be a two column matrix.

Author(s)

T. W. Yee

References

Reid, N. (2003) Asymptotics and the theory of inference. *Annals of Statistics*, **31**, 1695–1731.

See Also

exponential.

Examples

```
x  = runif(n <- 1000)
theta = exp(-2+x)
y1 = rexp(n, rate=exp(-theta)/theta)
y2 = 1 + rexp(n, rate=theta)
fit = vglm(cbind(y1,y2) ~ x, fam=gammahyp(expected=TRUE), trace=TRUE)
fit = vglm(cbind(y1,y2) ~ x, fam=gammahyp, trace=TRUE, crit="coef")
coef(fit, matrix=TRUE)
Coef(fit)
fitted(fit)[1:4,]
summary(fit)
```

---

garma *GARMA (Generalized Autoregressive Moving-Average) Models*

Description

Fits GARMA models to time series data.

Usage

```
garma(link = c("identity", "loge", "reciprocal",
               "logit", "probit", "cloglog", "cauchit"),
      earg=list(),
      p.ar.lag = 1, q.lag.ma = 0,
      coefstart = NULL, step = 1)
```

Arguments

link	Link function applied to the mean response. By default, the first choice is used, which is suitable for continuous responses. The link loge should be chosen if the data are counts. The links logit, probit, cloglog, cauchit are suitable for binary responses.
earg	List. Extra argument for the link. See earg in Links for general information. In particular, this argument is useful when the log or logit link is chosen: for log and logit, zero values can be replaced by bvalue which is inputted as earg=list(bvalue = bvalue). See loge and logit etc. for specific information about each link function.
p.ar.lag	A positive integer, the lag for the autoregressive component. Called p below.
q.lag.ma	A non-negative integer, the lag for the moving-average component. Called q below.
coefstart	Starting values for the coefficients. For technical reasons, the argument coefstart in vglm cannot be used.
step	Numeric. Step length, e.g., 0.5 means half-stepsizing.

Details

This function draws heavily on Benjamin *et al.* (1998). See also Benjamin *et al.* (2003). GARMA models extend the ARMA time series model to generalized responses in the exponential family, e.g., Poisson counts, binary responses. Currently, this function can handle continuous, count and binary responses only. The possible link functions given in the link argument reflect this, and the user must choose an appropriate link.

The GARMA(p, q) model is defined by firstly having a response belonging to the exponential family

$$f(y_t|D_t) = \exp\left\{\frac{y_t\theta_t - b(\theta_t)}{\phi/A_t} + c(y_t, \phi/A_t)\right\}$$

where θ_t and ϕ are the canonical and scale parameters respectively, and A_t are known prior weights. The mean $\mu_t = E(Y_t|D_t) = b'(\theta_t)$ is related to the linear predictor η_t by the link function g. Here, $D_t = \{x_t, \ldots, x_1, y_{t-1}, \ldots, y_1, \mu_{t-1}, \ldots, \mu_1\}$ is the previous information set. Secondly, the GARMA(p, q) model is defined by

$$g(\mu_t) = \eta_t = x_t^T\beta + \sum_{k=1}^{p}\phi_k(g(y_{t-k}) - x_{t-k}^T\beta) + \sum_{k=1}^{q}\theta_k(g(y_{t-k}) - \eta_{t-k}).$$

Parameter vectors β, ϕ and θ are estimated by maximum likelihood.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm.

Warning

This **VGAM** family function is 'non-standard' in that the model does need some coercing to get it into the VGLM framework. Special code is required to get it running. A consequence is that some methods functions may give wrong results when applied to the fitted object.

Note

This function is unpolished and is requires lots of improvements. In particular, initialization is quite poor, and could be improved. A limited amount of experience has shown that half-stepsizing is often needed for convergence, therefore choosing `crit="coef"` is not recommended.

Overdispersion is not handled.

Author(s)

T. W. Yee

References

Benjamin, M. A., Rigby, R. A. and Stasinopoulos, M. D. (1998) Fitting Non-Gaussian Time Series Models. Pages 191–196 in: *Proceedings in Computational Statistics COMPSTAT 1998* by Payne, R. and P. J. Green. Physica-Verlag.

Benjamin, M. A., Rigby, R. A. and Stasinopoulos, M. D. (2003) Generalized Autoregressive Moving Average Models. *Journal of the American Statistical Association*, **98**: 214–223.

Zeger, S. L. and Qaqish, B. (1988) Markov regression models for time series: a quasi-likelihood approach. *Biometrics*, **44**: 1019–1031.

See Also

The site `http://www.stat.auckland.ac.nz/~yee` contains more documentation about this family function.

Examples

```
# See Zeger and Qaqish (1988)
interspike = c(68, 41, 82, 66, 101, 66, 57,  41,  27, 78,
59, 73,  6, 44,  72, 66, 59,  60,  39, 52,
50, 29, 30, 56,  76, 55, 73, 104, 104, 52,
25, 33, 20, 60,  47,  6, 47,  22,  35, 30,
29, 58, 24, 34,  36, 34,  6,  19,  28, 16,
36, 33, 12, 26,  36, 39, 24,  14,  28, 13,
 2, 30, 18, 17,  28,  9, 28,  20,  17, 12,
19, 18, 14, 23,  18, 22, 18,  19,  26, 27,
23, 24, 35, 22,  29, 28, 17,  30,  34, 17,
20, 49, 29, 35,  49, 25, 55,  42,  29, 16)
spikenum = seq(interspike)
bvalue = 0.1  # .Machine$double.xmin # Boundary value
fit = vglm(interspike ~ 1, trace=TRUE,
   garma("loge", earg=list(bvalue=bvalue), p=2, coef=c(4,.3,.4)))
summary(fit)
coef(fit, matrix=TRUE)
Coef(fit)  # A bug here
## Not run:
plot(interspike, ylim=c(0,120), las=1, font=1, xlab="Spike Number",
     ylab="Inter-Spike Time (ms)", col="blue")
lines(spikenum[-(1:fit@misc$plag)], fitted(fit), col="green")
abline(h=mean(interspike), lty=2)
```

```
## End(Not run)
```

`gaussianff` *Gaussian (normal) Family Function*

Description

Fits a generalized linear model to a response with Gaussian (normal) errors. The dispersion parameter may be known or unknown.

Usage

```
gaussianff(dispersion = 0, parallel = FALSE, zero = NULL)
```

Arguments

`parallel`	A logical or formula. If a formula, the response of the formula should be a logical and the terms of the formula indicates whether or not those terms are parallel.
`dispersion`	Dispersion parameter. If 0 then it is estimated and the moment estimate is put in `object@misc$dispersion`; it is assigned the value $$\sum_{i=1}^{n} (y_i - \eta_i)^T W_i (y_i - \eta_i)/(nM - p)$$ where p is the total number of parameters estimated (for RR-VGLMs the value used is the number of columns in the large X model matrix; this may not be correct). If the argument is assigned a positive quantity then it is assumed to be known with that value.
`zero`	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,...,M} where M is the number of columns of the matrix response.

Details

This function is usually used in conjunction with `vglm`, else `vlm` is recommended instead. The notation M is used to denote the number of linear/additive predictors. This function can handle any finite M, and the default is to use ordinary least squares. A vector linear/additive model can be fitted by minimizing

$$\sum_{i=1}^{n} (y_i - \eta_i)^T W_i (y_i - \eta_i)$$

where y_i is a M-vector, η_i is the vector of linear/additive predictors. The W_i is any positive-definite matrix, and the default is the order-M identity matrix. The W_i can be inputted using the `weights` argument of `vlm`/`vglm`/`vgam` etc., and the format is the *matrix-band* format whereby it is a $n \times A$ matrix with the diagonals are passed first, followed by next the upper band, all the way to the $(1, M)$ element. Here, A has maximum value of $M(M+1)/2$ and a minimum value of M. Usually the `weights` argument of `vlm`/`vglm`/`vgam`/`rrvglm` is just a vector, in which case each element is multiplied by a order-M identity matrix. If in doubt, type something like `weights(object, type="working")` after the model has been fitted.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

Note

This **VGAM** family function is supposed to be similar to `gaussian` but is is not compatible with `glm`. The `"ff"` in the name is added to avoid any masking problems.

Author(s)

Thomas W. Yee

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

Yee, T. W. and Wild, C. J. (1996) Vector generalized additive models. *Journal of the Royal Statistical Society, Series B, Methodological*, **58**, 481–493.

See Also

`normal1`, `vlm`, `vglm`, `vgam`, `rrvglm`.

Examples

```
d = data.frame(x = sort(runif(n <- 40)))
d = transform(d, y1 = 1 + 2*x + rnorm(n, sd=0.1),
                 y2 = 3 + 4*x + rnorm(n, sd=0.1),
                 y3 = 7 + 4*x + rnorm(n, sd=0.1))
fit = vglm(cbind(y1,y2) ~ x, gaussianff, data=d)
coef(fit, matrix=TRUE)

# For comparison:
coef( lmfit <-  lm(y1 ~ x, data=d))
coef(glmfit <- glm(y2 ~ x, data=d, gaussian))
vcov(fit)
vcov(lmfit)

t(weights(fit, type="prior"))          # Unweighted observations
weights(fit, type="working")[1:4,]     # Identity matrices

# Reduced-rank VLM (rank-1)
fit2 = rrvglm(cbind(y1,y2,y3) ~ x, gaussianff, data=d)
Coef(fit2)
```

genbetaII *Generalized Beta Distribution of the Second Kind*

Description

Maximum likelihood estimation of the 4-parameter generalized beta II distribution.

Usage

```
genbetaII(link.a = "loge", link.scale = "loge",
          link.p = "loge", link.q = "loge",
          earg.a=list(), earg.scale=list(), earg.p=list(), earg.q=list(),
          init.a = NULL, init.scale = NULL, init.p = 1, init.q = 1,
          zero = NULL)
```

Arguments

`link.a, link.scale, link.p, link.q`
: Parameter link functions applied to the shape parameter `a`, scale parameter `scale`, shape parameter `p`, and shape parameter `q`. All four parameters are positive. See `Links` for more choices.

`earg.a, earg.scale, earg.p, earg.q`
: List. Extra argument for each of the links. See `earg` in `Links` for general information.

`init.a, init.scale`
: Optional initial values for `a` and `scale`. A `NULL` means a value is computed internally.

`init.p, init.q`
: Optional initial values for `p` and `q`.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. Here, the values must be from the set {1,2,3,4} which correspond to `a`, `scale`, `p`, `q`, respectively.

Details

This distribution is most useful for unifying a substantial number of size distributions. For example, the Singh-Maddala, Dagum, Fisk (log-logistic), Lomax (Pareto type II), inverse Lomax, beta distribution of the second kind distributions are all special cases. Full details can be found in Kleiber and Kotz (2003), and Brazauskas (2002).

The 4-parameter generalized beta II distribution has density

$$f(y) = ay^{ap-1}/[b^{ap}B(p,q)\{1+(y/b)^a\}^{p+q}]$$

for $a > 0$, $b > 0$, $p > 0$, $q > 0$, $y > 0$. Here B is the beta function, and b is the scale parameter `scale`, while the others are shape parameters. The mean is

$$E(Y) = b\,\Gamma(p+1/a)\,\Gamma(q-1/a)/(\Gamma(p)\,\Gamma(q))$$

provided $-ap < 1 < aq$.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

If the self-starting initial values fail, try experimenting with the initial value arguments, especially those whose default value is not NULL.

Successful convergence depends on having very good initial values. This is rather difficult for this distribution! More improvements could be made here.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

Brazauskas, V. (2002) Fisher information matrix for the Feller-Pareto distribution. *Statistics & Probability Letters*, **59**, 159–167.

See Also

lino, betaff, betaII, dagum, sinmad, fisk, lomax, invlomax, paralogistic, invparalogistic.

Examples

```
y = rsinmad(n=3000, 4, 6, 2) # Not very good data!
fit = vglm(y ~ 1, genbetaII, trace=TRUE)
fit = vglm(y ~ 1, genbetaII(init.p=1.0, init.a=4, init.sc=7, init.q=2.3),
           trace=TRUE, crit="c")
coef(fit, mat=TRUE)
Coef(fit)
summary(fit)
```

---

genpoisson *Generalized Poisson distribution*

Description

Estimation of the two parameters of a generalized Poisson distribution.

Usage

```
genpoisson(llambda = "logit", ltheta = "loge",
           elambda=list(), etheta=list(),
           ilambda = 0.5, itheta = NULL, zero = NULL)
```

Arguments

llambda	Parameter link function for lambda. See Links for more choices. The parameter lies in the unit interval, therefore the logit link is a reasonable default.
ltheta	Parameter link function for theta. See Links for more choices. The parameter is positive, therefore the default is the log link.
elambda, etheta	List. Extra argument for each of the links. See earg in Links for general information.
ilambda	Optional initial value for lambda.
itheta	Optional initial value for theta.
zero	An integer vector, containing the value 1 or 2. If so, lambda or theta respectively are modelled as an intercept only. The default value NULL means both linear/additive predictors are modelled as functions of the explanatory variables.

Details

See Consul (1989) for details.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

Convergence problems may occur when lambda is very close to 0. This distribution is useful for dispersion modelling.

Author(s)

T. W. Yee

References

Jorgensen, B. (1997) *The Theory of Dispersion Models*. London: Chapman & Hall

Consul, P. C. (1989) *Generalized Poisson Distributions: Properties and Applications*, Marcel Dekker.

See Also

poissonff.

Examples

```
y = rpois(n=100, lam=24)
fit  = vglm(y ~ 1, genpoisson, trace=TRUE)
fitted(fit)[1:5]
mean(y)
summary(fit)
```

```
coef(fit, matrix=TRUE)
Coef(fit)
```

`geometric` *Geometric Distribution*

Description

Maximum likelihood estimation for the geometric distribution.

Usage

```
geometric(link = "logit", earg=list(), expected = TRUE)
```

Arguments

`link`	Parameter link function applied to the parameter p, which lies in the unit interval. See `Links` for more choices.
`earg`	List. Extra argument for the link. See `earg` in `Links` for general information.
`expected`	Logical. Fisher scoring is used if `expected = TRUE`, else Newton-Raphson.

Details

A random variable Y has a 1-parameter geometric distribution if $P(Y = y) = p(1-p)^y$ for $y = 0, 1, 2, \ldots$. Here, p is the probability of success, and Y is the number of (independent) trials that are fails until a success occurs. Thus the response Y should be a non-negative integer. The mean of Y is $E(Y) = (1-p)/p$ and its variance is $Var(Y) = (1-p)/p^2$. The geometric distribution is a special case of the negative binomial distribution (see `negbinomial`).

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Author(s)

T. W. Yee

References

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

See Also

`negbinomial`, `Geometric`, `betageometric`, `rbetageom`.

Examples

```
x1 = runif(n <- 1000) - 0.5
x2 = runif(n) - 0.5
x3 = runif(n) - 0.5
eta = 0.2 - 0.7 * x1 + 1.9 * x2
prob = logit(eta, inverse=TRUE)
y = rgeom(n, prob)
table(y)
fit = vglm(y ~ x1 + x2 + x3, geometric, trace=TRUE, crit="coef")
coef(fit)
coef(fit, mat=TRUE)
summary(fit)
```

---

get.smart *Retrieve One Component of ".smart.prediction"*

Description

Retrieve one component of the list `.smart.prediction` from `smartpredenv` (R) or frame 1 (S-PLUS).

Usage

```
get.smart()
```

Details

`get.smart` is used in `"read"` mode within a smart function: it retrieves parameters saved at the time of fitting, and is used for prediction. `get.smart` is only used in smart functions such as `poly`; `get.smart.prediction` is only used in modelling functions such as `lm` and `glm`. The function `get.smart` gets only a part of `.smart.prediction` whereas `get.smart.prediction` gets the entire `.smart.prediction`.

Value

Returns with one list component of `.smart.prediction` from `smartpredenv` (R) or frame 1 (S-PLUS), in fact, `.smart.prediction[[.smart.prediction.counter]]`. The whole procedure mimics a first-in first-out stack (better known as a *queue*).

Side Effects

The variable `.smart.prediction.counter` in `smartpredenv` (R) or frame 1 (S-PLUS) is incremented beforehand, and then written back to `smartpredenv` (R) or frame 1 (S-PLUS).

See Also

`get.smart.prediction`.

Examples

```
"my1" <- function(x, minx=min(x)) { # Here is a smart function
    x <- x  # Needed for nested calls, e.g., bs(scale(x))
    if(smart.mode.is("read")) {
        smart  <- get.smart()
        minx <- smart$minx          # Overwrite its value
    } else
    if(smart.mode.is("write"))
        put.smart(list(minx=minx))
    sqrt(x-minx)
}
attr(my1, "smart") <- TRUE
```

---

get.smart.prediction

Retrieves ".smart.prediction"

Description

Retrieves `.smart.prediction` from `smartpredenv` (R) or frame 1 (S-PLUS).

Usage

```
get.smart.prediction()
```

Details

A smart modelling function such as `lm` allows smart functions such as `bs` to write to a data structure called `.smart.prediction` in `smartpredenv` (R) or frame 1 (S-PLUS). At the end of fitting, `get.smart.prediction` retrieves this data structure. It is then attached to the object, and used for prediction later.

Value

Returns with the list `.smart.prediction` from `smartpredenv` (R) or frame 1 (S-PLUS).

See Also

`get.smart`, `lm`.

Examples

```
## Not run:
# Put at the end of lm
fit$smart <- get.smart.prediction()
## End(Not run)
```

gev *Generalized Extreme Value Distribution Family Function*

Description

Maximum likelihood estimation of the 3-parameter generalized extreme value (GEV) distribution.

Usage

```
gev(llocation = "identity", lscale = "loge", lshape = "logoff",
    elocation = list(), escale = list(),
    eshape = if(lshape=="logoff") list(offset=0.5) else
    if(lshape=="elogit") list(min=-0.5, max=0.5) else list(),
    percentiles = c(95, 99),
    iscale=NULL, ishape = NULL,
    method.init = 1, gshape=c(-0.45, 0.45), tshape0=0.001, zero = 3)
egev(llocation = "identity", lscale = "loge", lshape = "logoff",
     elocation = list(), escale = list(),
     eshape = if(lshape=="logoff") list(offset=0.5) else
     if(lshape=="elogit") list(min=-0.5, max=0.5) else list(),
     percentiles = c(95, 99),
     iscale=NULL,  ishape = NULL,
     method.init=1, gshape=c(-0.45, 0.45), tshape0=0.001, zero = 3)
```

Arguments

`llocation, lscale, lshape`
: Parameter link functions for μ, σ and ξ respectively. See `Links` for more choices.

`elocation, escale, eshape`
: List. Extra argument for the respective links. See `earg` in `Links` for general information. For the shape parameter, if the `logoff` link is chosen then the offset is called A below; and then the linear/additive predictor is $\log(\xi + A)$ which means that $\xi > -A$. For technical reasons (see **Details**) it is a good idea for $A = 0.5$.

`percentiles`
: Numeric vector of percentiles used for the fitted values. Values should be between 0 and 100. However, if `percentiles=NULL`, then the mean $\mu + \sigma(\Gamma(1-\xi)-1)/\xi$ is returned, and this is only defined if $\xi < 1$.

`iscale, ishape`
: Numeric. Initial value for σ and ξ. A `NULL` means a value is computed internally. The argument `ishape` is more important than the other two because they are initialized from the initial ξ. If a failure to converge occurs, or even to obtain initial values occurs, try assigning `ishape` some value (positive or negative; the sign can be very important). Also, in general, a larger value of `iscale` is better than a smaller value.

`method.init` Initialization method. Either the value 1 or 2. Method 1 involves choosing the best ξ on a course grid with endpoints `gshape`. Method 2 is similar to the method of moments. If both methods fail try using `ishape`.

`gshape` Numeric, of length 2. Range of ξ used for a grid search for a good initial value for ξ. Used only if `method.init` equals 1.

`tshape0` Positive numeric. Threshold/tolerance value for resting whether ξ is zero. If the absolute value of the estimate of ξ is less than this value then it will be assumed zero and Gumbel derivatives etc. will be used.

`zero` An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,3} corresponding respectively to μ, σ, ξ. If `zero=NULL` then all linear/additive predictors are modelled as a linear combination of the explanatory variables. For many data sets having `zero=3` is a good idea.

Details

The GEV distribution function can be written

$$G(y) = \exp(-[(y-\mu)/\sigma]_+^{-1/\xi})$$

where $\sigma > 0$, $-\infty < \mu < \infty$, and $1 + \xi(y-\mu)/\sigma > 0$. Here, $x_+ = \max(x, 0)$. The μ, σ, ξ are known as the *location*, *scale* and *shape* parameters respectively. The cases $\xi > 0$, $\xi < 0$, $\xi = 0$ correspond to the Frechet, Weibull, and Gumbel types respectively. It can be noted that the Gumbel (or Type I) distribution accommodates many commonly-used distributions such as the normal, lognormal, logistic, gamma, exponential and Weibull.

For the GEV distribution, the kth moment about the mean exists if $\xi < 1/k$. Provided they exist, the mean and variance are given by $\mu + \sigma\{\Gamma(1-\xi) - 1\}/\xi$ and $\sigma^2\{\Gamma(1-2\xi) - \Gamma^2(1-\xi)\}/\xi^2$ respectively, where Γ is the gamma function.

Smith (1985) established that when $\xi > -0.5$, the maximum likelihood estimators are completely regular. To have some control over the estimated ξ try using `lshape="logoff"` and the `eshape=list(offset=0.5)`, say, or `lshape="elogit"` and `eshape=list(min=-0.5, max=0.5)`, say.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Warning

Currently, if an estimate of ξ is too close to zero then an error will occur for `gev()` with multivariate responses. In general, `egev()` is more reliable than `gev()`.

Fitting the GEV by maximum likelihood estimation can be numerically fraught. If $1+\xi(y-\mu)/\sigma \leq 0$ then some crude evasive action is taken but the estimation process can still fail. This is particularly the case if `vgam` with `s` is used; then smoothing is best done with `vglm` with regression splines (`bs` or `ns`) because `vglm` implements half-stepsizing whereas `vgam` doesn't (half-stepsizing helps handle the problem of straying outside the parameter space.)

Note

The **VGAM** family function `gev` can handle a multivariate (matrix) response. If so, each row of the matrix is sorted into descending order and `NA`s are put last. With a vector or one-column matrix response using `egev` will give the same result but be faster and it handles the $\xi = 0$ case. The function `gev` implements Tawn (1988) while `egev` implements Prescott and Walden (1980).

The shape parameter ξ is difficult to estimate accurately unless there is a lot of data. Convergence is slow when ξ is near -0.5. Given many explanatory variables, it is often a good idea to make sure `zero=3`. The range restrictions of the parameter ξ are not enforced; thus it is possible for a violation to occur.

Successful convergence often depends on having a reasonably good initial value for ξ. If failure occurs try various values for the argument `ishape`, and if there are covariates, having `zero=3` is advised.

Author(s)

T. W. Yee

References

Yee, T. W. and Stephenson, A. G. (2007) Vector generalized linear and additive extreme value models. *Extremes*, **10**, 1–19.

Tawn, J. A. (1988) An extreme-value theory model for dependent observations. *Journal of Hydrology*, **101**, 227–250.

Prescott, P. and Walden, A. T. (1980) Maximum likelihood estimation of the parameters of the generalized extreme-value distribution. *Biometrika*, **67**, 723–724.

Smith, R. L. (1985) Maximum likelihood estimation in a class of nonregular cases. *Biometrika*, **72**, 67–90.

See Also

`rgev`, `gumbel`, `egumbel`, `guplot`, `rlplot.egev`, `gpd`, `elogit`, `oxtemp`, `venice`.

Examples

```
# Multivariate example
data(venice)
y = as.matrix(venice[,paste("r", 1:10, sep="")])
fit1 = vgam(y[,1:2] ~ s(year, df=3), gev(zero=2:3), venice, trace=TRUE)
coef(fit1, matrix=TRUE)
fitted(fit1)[1:4,]
## Not run:
par(mfrow=c(1,2), las=1)
plot(fit1, se=TRUE, lcol="blue", scol="forestgreen",
     main="Fitted mu(year) function (centered)")
attach(venice)
matplot(year, y[,1:2], ylab="Sea level (cm)", col=1:2,
        main="Highest 2 annual sealevels and fitted 95 percentile")
lines(year, fitted(fit1)[,1], lty="dashed", col="blue")
```

```
detach(venice)
## End(Not run)

# Univariate example
data(oxtemp)
(fit = vglm(maxtemp ~ 1, egev, data=oxtemp, trace=TRUE))
fitted(fit)[1:3,]
coef(fit, mat=TRUE)
Coef(fit)
vcov(fit)
vcov(fit, untransform=TRUE)
sqrt(diag(vcov(fit)))   # Approximate standard errors
## Not run:  rlplot(fit)
```

`gevUC` *The Generalized Extreme Value Distribution*

Description

Density, distribution function, quantile function and random generation for the generalized extreme value distribution (GEV) with location parameter `location`, scale parameter `scale` and shape parameter `shape`.

Usage

```
dgev(x, location=0, scale=1, shape=0)
pgev(q, location=0, scale=1, shape=0)
qgev(p, location=0, scale=1, shape=0)
rgev(n, location=0, scale=1, shape=0)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. Positive integer of length 1.
`location`	the location parameter μ.
`scale`	the scale parameter σ. Must consist of positive values.
`shape`	the shape parameter ξ.

Details

See `gev`, the **VGAM** family function for estimating the two parameters by maximum likelihood estimation, for formulae and other details. Apart from `n`, all the above arguments may be vectors and are recyled to the appropriate length if necessary.

Value

dgev gives the density, pgev gives the distribution function, qgev gives the quantile function, and rgev generates random deviates.

Note

The default value of $\xi = 0$ means the default distribution is the Gumbel.

Currently, these functions have different argument names compared with those in the **evd** package.

Author(s)

T. W. Yee

References

Coles, S. (2001) *An Introduction to Statistical Modeling of Extreme Values*. London: Springer-Verlag.

See Also

gev.

Examples

```
## Not run:
x = seq(-3, 3, by=0.01)
loc = 0; sigma = 1; xi = -0.4
plot(x, dgev(x, loc, sigma, xi), type="l", col="blue", ylim=c(0,1),
     main="Blue is density, red is cumulative distribution function",
     sub="Purple are 5,10,...,95 percentiles", ylab="", las=1)
abline(h=0, col="blue", lty=2)
lines(qgev(seq(0.05,0.95,by=0.05), loc, sigma, xi),
      dgev(qgev(seq(0.05,0.95,by=0.05), loc, sigma, xi), loc, sigma, xi),
      col="purple", lty=3, type="h")
lines(x, pgev(x, loc, sigma, xi), type="l", col="red")
abline(h=0, lty=2)

pgev(qgev(seq(0.05,0.95,by=0.05), loc, sigma, xi), loc, sigma, xi)
## End(Not run)
```

---

gew *General Electric and Westinghouse Data*

Description

General Electric and Westinghouse capital data.

Usage

```
data(gew)
```

Format

A data frame with 20 observations on the following 6 variables.

y1 a numeric vector which may be regarded as investment figures for the two companies

x1 market values

x2 capital stocks

y2 a numeric vector which may be regarded as investment figures for the two companies

x3 market values

x4 capital stocks

Details

The period is 1934 to 1953.

Source

Unknown.

References

Zellner, A. (1962) An efficient method of estimating seemingly unrelated regressions and tests for aggregation bias. *Journal of the American Statistical Association*, **57**, 348–368.

Examples

```
data(gew)
str(gew)
```

---

ggamma *Generalized Gamma distribution family function*

Description

Estimation of the 3-parameter generalized gamma distribution proposed by Stacy (1962).

Usage

```
ggamma(lscale="loge", ld="loge", lk="loge",
       escale=list(), ed=list(), ek=list(),
       iscale=NULL, id=NULL, ik=NULL, zero=NULL)
```

Arguments

`lscale, ld, lk`
: Parameter link function applied to each of the positive parameters b, d and k, respectively. See `Links` for more choices.

`escale, ed, ek`
: List. Extra argument for each of the links. See `earg` in `Links` for general information.

`iscale, id, ik`
: Initial value for b, d and k, respectively. The defaults mean an initial value is determined internally for each.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,3}. The default value means none are modelled as intercept-only terms.

Details

The probability density function can be written

$$f(y; b, d, k) = db^{-dk} y^{dk-1} \exp[-(y/b)^d]/\Gamma(k)$$

for scale parameter $b > 0$, and $d > 0$, $k > 0$, and $y > 0$. The mean of Y is bk (returned as the fitted values).

There are many special cases, as given in Table 1 of Stacey and Mihram (1965). In the following, the parameters are in the order b, d, k. The special cases are: Exponential $f(y; b, 1, 1)$, Gamma $f(y; b, 1, k)$, Weibull $f(y; b, d, 1)$, Chi Squared $f(y; 2, 1, a/2)$ with a degrees of freedom, Chi $f(y; \sqrt{2}, 2, a/2)$ with a degrees of freedom, Half-normal $f(y; \sqrt{2}, 2, 1/2)$, Circular normal $f(y; \sqrt{2}, 2, 1)$, Spherical normal $f(y; \sqrt{2}, 2, 3/2)$, Rayleigh $f(y; c\sqrt{2}, 2, 1)$ where $c > 0$.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Warning

Several authors have considered maximum likelihood estimation for the generalized gamma distribution and have found that the Newton-Raphson algorithm does not work very well and that the existence of solutions to the log-likelihood equations is sometimes in doubt. Although Fisher scoring is used here, it is likely that the same problems will be encountered. It appears that large samples are required, for example, the estimator of k became asymptotically normal only with 400 or more observations. It is not uncommon for maximum likelihood estimates to fail to converge even with two or three hundred observations. With covariates, even more observations are needed to increase the chances of convergence.

Note

The notation used here differs from Stacy (1962) and Prentice (1974). Poor initial values may result in failure to converge so if there are covariates and there are convergence problems, try using the `zero` argument (e.g., `zero=2:3`) or the `ik` argument.

Author(s)

T. W. Yee

References

Stacy, E. W. (1962) A generalization of the gamma distribution. *Annals of Mathematical Statistics*, **33**, 1187–1192.

Stacy, E. W. and Mihram, G. A. (1965) Parameter estimation for a generalized gamma distribution. *Technometrics*, **7**, 349–358.

Prentice, R. L. (1974) A log gamma model and its maximum likelihood estimation. *Biometrika*, **61**, 539–544.

See Also

`rggamma`, `gamma1`, `gamma2`, `prentice74`.

Examples

```
k = exp(-1)
Scale = exp(1)
y = rgamma(n=1000, shape=k, scale=Scale)
fit = vglm(y ~ 1, ggamma, trace=TRUE)
coef(fit, matrix=TRUE)

# Another example
x = runif(n <- 5000)
Scale = exp(1)
d = exp(0 + 1.2*x)
k = exp(-1 + 2*x)
y = rggamma(n, scale=Scale, d=d, k=k)
fit = vglm(y ~ x, ggamma(zero=1, iscal=6), trace=TRUE)
coef(fit, matrix=TRUE)
```

---

ggammaUC *The Generalized Gamma Distribution*

Description

Density, distribution function, quantile function and random generation for the generalized gamma distribution with scale parameter `scale`, and parameters `d` and `k`.

Usage

```
dggamma(x, scale=1, d=1, k=1)
pggamma(q, scale=1, d=1, k=1)
qggamma(p, scale=1, d=1, k=1)
rggamma(n, scale=1, d=1, k=1)
```

Arguments

x, q	vector of quantiles.
p	vector of probabilities.
n	number of observations. Positive integer of length 1.
scale	the (positive) scale parameter b.
d, k	the (positive) parameters d and k.

Details

See ggamma, the **VGAM** family function for estimating the generalized gamma distribution by maximum likelihood estimation, for formulae and other details. Apart from n, all the above arguments may be vectors and are recyled to the appropriate length if necessary.

Value

dggamma gives the density, pggamma gives the distribution function, qggamma gives the quantile function, and rggamma generates random deviates.

Author(s)

T. W. Yee

References

Stacy, E. W. and Mihram, G. A. (1965) Parameter estimation for a generalized gamma distribution. *Technometrics*, **7**, 349–358.

See Also

ggamma.

Examples

```
## Not run:
x=seq(0, 14, by=0.01); d=1.5; Scale=2; k=6
plot(x, dggamma(x, Scale, d, k), type="l", col="blue", ylim=c(0,1),
     main="Blue is density, red is cumulative distribution function",
     sub="Purple are 5,10,...,95 percentiles", las=1, ylab="")
abline(h=0, col="blue", lty=2)
lines(qggamma(seq(0.05,0.95,by=0.05), Scale, d, k),
      dggamma(qggamma(seq(0.05,0.95,by=0.05), Scale, d, k), Scale, d, k),
      col="purple", lty=3, type="h")
lines(x, pggamma(x, Scale, d, k), type="l", col="red")
abline(h=0, lty=2)
## End(Not run)
```

golf *Gamma-Ordinal Link Function*

Description

Computes the gamma-ordinal transformation, including its inverse and the first two derivatives.

Usage

```
golf(theta, earg = list(lambda = 1), inverse = FALSE, deriv = 0,
     short = TRUE, tag = FALSE)
```

Arguments

theta	Numeric or character. See below for further details.
earg	Extra argument for passing in additional information. This must be list with component lambda. Here, lambda is the shape parameter in gamma2. A component in the list called cutpoint is optional; if omitted then cutpoint is ignored from the GOLF definition. If given, the cutpoints should be non-negative integers. If golf() is used as the link function in cumulative then, if the cutpoints are known, then one should choose reverse=TRUE, parallel=TRUE, intercept.apply=TRUE. If the cutpoints are unknown, then choose reverse=TRUE, parallel=TRUE, intercept.apply=FALSE.
inverse	Logical. If TRUE the inverse function is computed.
deriv	Order of the derivative. Integer with value 0, 1 or 2.
short	Used for labelling the blurb slot of a vglmff-class object.
tag	Used for labelling the linear/additive predictor in the initialize slot of a vglmff-class object. Contains a little more information if TRUE.

Details

The gamma-ordinal link function (GOLF) can be applied to a parameter lying in the unit interval. Its purpose is to link cumulative probabilities associated with an ordinal response coming from an underlying 2-parameter gamma distribution.

The arguments short and tag are used only if theta is character.

See Links for general information about **VGAM** link functions.

Value

See Yee (2006) for details.

Warning

Prediction may not work on vglm or vgam etc. objects if this link function is used.

Note

Numerical values of `theta` too close to 0 or 1 or out of range result in large positive or negative values, or maybe 0 depending on the arguments. Although measures have been taken to handle cases where `theta` is too close to 1 or 0, numerical instabilities may still arise.

In terms of the threshold approach with cumulative probabilities for an ordinal response this link function corresponds to the gamma distribution (see `gamma2`) that has been recorded as an ordinal response using known cutpoints.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2006) *Ordinal ordination with normalizing link functions for count data*, (submitted for publication).

See Also

`Links`, `gamma2`, `polf`, `nbolf`, `cumulative`.

Examples

```
earg = list(lambda=1)
golf("p", earg=earg, short=FALSE)
golf("p", earg=earg, tag=TRUE)

p = seq(0.02, 0.98, len=201)
y = golf(p, earg=earg)
y. = golf(p, earg=earg, deriv=1)
max(abs(golf(y, earg=earg, inv=TRUE) - p)) # Should be 0

## Not run:
par(mfrow=c(2,1), las=1)
plot(p, y, type="l", col="blue", main="golf()")
abline(h=0, v=0.5, col="red", lty="dashed")

plot(p, y., type="l", col="blue",
     main="(Reciprocal of) first GOLF derivative")
## End(Not run)

# Another example
nn = 1000
x2 = sort(runif(nn))
x3 = runif(nn)
mymu = exp( 3 + 1 * x2 - 2 * x3)
lambda = 4
y1 = rgamma(nn, shape=lambda, scale=mymu/lambda)
cutpoints = c(-Inf, 10, 20, Inf)
cuty = Cut(y1, breaks=cutpoints)
## Not run:
```

```
par(mfrow=c(1,1), las=1)
plot(x2, x3, col=cuty, pch=as.character(cuty))
## End(Not run)
table(cuty) / sum(table(cuty))
fit = vglm(cuty ~ x2 + x3, fam = cumulative(link="golf",
           reverse=TRUE, parallel=TRUE, intercept.apply=TRUE,
           mv=TRUE, earg=list(cutpoint=cutpoints[2:3], lambda=lambda)),
           trace=TRUE)
fit@y[1:5,]
fitted(fit)[1:5,]
predict(fit)[1:5,]
coef(fit)
coef(fit, matrix=TRUE)
constraints(fit)
fit@misc$earg
```

`gpd` *Generalized Pareto Distribution Family Function*

Description

Maximum likelihood estimation of the 2-parameter generalized Pareto distribution (GPD).

Usage

```
gpd(threshold = 0, lscale = "loge", lshape = "logoff",
    escale = list(),
    eshape = if(lshape=="logoff") list(offset=0.5) else
                if(lshape=="elogit") list(min=-0.5, max=0.5) else NULL,
    percentiles = c(90, 95), iscale = NULL, ishape = NULL,
    tshape0=0.001, method.init=1, zero=2)
```

Arguments

`threshold` Numeric, values are recycled if necessary. The threshold value(s), called μ below.

`lscale` Parameter link function for the scale parameter σ. See `Links` for more choices.

`lshape` Parameter link function for the shape parameter ξ. See `Links` for more choices. The default constrains the parameter to be greater than -0.5 because if $\xi \leq -0.5$ then Fisher scoring does not work. See the Details section below for more information.

`escale, eshape` Extra argument for the `lscale` and `lshape` arguments. See `earg` in `Links` for general information. For the shape parameter, if the `logoff` link is chosen then the offset is called A below; and then the second linear/additive predictor is $\log(\xi + A)$ which means that $\xi > -A$. The working weight matrices are positive definite if $A = 0.5$.

`percentiles` Numeric vector of percentiles used for the fitted values. Values should be between 0 and 100. See the example below for illustration. However, if `percentiles=NULL` then the mean $\mu + \sigma/(1-\xi)$ is returned; this is only defined if $\xi < 1$.

`iscale, ishape` Numeric. Optional initial values for σ and ξ. The default is to use `method.init` and compute a value internally for each parameter. Values of `ishape` should be between -0.5 and 1. Values of `iscale` should be positive.

`tshape0` Positive numeric. Threshold/tolerance value for resting whether ξ is zero. If the absolute value of the estimate of ξ is less than this value then it will be assumed zero and exponential distribution derivatives etc. will be used.

`method.init` Method of initialization, either 1 or 2. The first is the method of moments, and the second is a variant of this. If neither work, try assigning values to arguments `ishape` and/or `iscale`.

`zero` An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The value must be from the set {1,2} corresponding respectively to σ and ξ. It is often a good idea for the σ parameter only to be modelled through a linear combination of the explanatory variables because the shape parameter is probably best left as an intercept only: `zero=2`. Setting `zero=NULL` means both parameters are modelled with explanatory variables.

Details

The distribution function of the GPD can be written

$$G(y) = 1 - [1 + \xi(y-\mu)/\sigma]_+^{-1/\xi}$$

where μ is the location parameter (known, with value `threshold`), $\sigma > 0$ is the scale parameter, ξ is the shape parameter, and $h_+ = \max(h, 0)$. The function $1 - G$ is known as the *survivor function*. The limit $\xi \to 0$ gives the *shifted exponential* as a special case:

$$G(y) = 1 - \exp[-(y-\mu)/\sigma].$$

The support is $y > \mu$ for $\xi > 0$, and $\mu < y < \mu - \sigma/\xi$ for $\xi < 0$.

Smith (1985) showed that if $\xi <= -0.5$ then this is known as the nonregular case and problems/difficulties can arise both theoretically and numerically. For the (regular) case $\xi > -0.5$ the classical asymptotic theory of maximum likelihood estimators is applicable; this is the default.

Although for $\xi < -0.5$ the usual asymptotic properties do not apply, the maximum likelihood estimator generally exists and is superefficient for $-1 < \xi < -0.5$, so it is "better" than normal. When $\xi < -1$ the maximum likelihood estimator generally does not exist as it effectively becomes a two parameter problem.

The mean of Y does not exist unless $\xi < 1$, and the variance does not exist unless $\xi < 0.5$. So if you want to fit a model with finite variance use `lshape="elogit"`.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`. However, for this **VGAM** family function, `vglm` is probably preferred over `vgam` when there is smoothing.

Warning

Fitting the GPD by maximum likelihood estimation can be numerically fraught. If $1+\xi(y-\mu)/\sigma \leq 0$ then some crude evasive action is taken but the estimation process can still fail. This is particularly the case if `vgam` with `s` is used. Then smoothing is best done with `vglm` with regression splines (`bs` or `ns`) because `vglm` implements half-stepsizing whereas `vgam` doesn't. Half-stepsizing helps handle the problem of straying outside the parameter space.

Note

The response in the formula of `vglm` and `vgam` is y. Internally, $y - \mu$ is computed.

With functions `rgpd`, `dgpd`, etc., the argument `location` matches with the argument `threshold` here.

Author(s)

T. W. Yee

References

Yee, T. W. and Stephenson, A. G. (2007) Vector generalized linear and additive extreme value models. *Extremes*, **10**, 1–19.

Coles, S. (2001) *An Introduction to Statistical Modeling of Extreme Values*. London: Springer-Verlag.

Smith, R. L. (1985) Maximum likelihood estimation in a class of nonregular cases. *Biometrika*, **72**, 67–90.

See Also

`rgpd`, `meplot`, `gev`, `pareto1`, `vglm`, `vgam`, `s`.

Examples

```
# Simulated data from an exponential distribution (xi=0)
threshold = 0.5
y = threshold + rexp(n=3000, rate=2)
fit = vglm(y ~ 1, gpd(threshold=threshold), trace=TRUE)
fitted(fit)[1:5,]
coef(fit, matrix=TRUE)    # xi should be close to 0
Coef(fit)
summary(fit)

fit@extra$threshold  # Note the threshold is stored here

# Check the 90 percentile
i = fit@y < fitted(fit)[1,"90%"]
100*table(i)/sum(table(i))    # Should be 90

# Check the 95 percentile
i = fit@y < fitted(fit)[1,"95%"]
100*table(i)/sum(table(i))    # Should be 95
```

```
## Not run:
plot(fit@y, col="blue", las=1, main="Fitted 90% and 95% quantiles")
matlines(1:length(fit@y), fitted(fit), lty=2:3, lwd=2)
## End(Not run)

# Another example
nn = 2000; threshold = 0; x = runif(nn)
xi = exp(-0.8)-0.5
y = rgpd(nn, scale=exp(1+0.2*x), shape=xi)
fit = vglm(y ~ x, gpd(threshold), trace=TRUE)
coef(fit, matrix=TRUE)

## Not run:
# Nonparametric fits
yy = y + rnorm(nn, sd=0.1)
fit1 = vgam(yy ~ s(x), gpd(threshold), trace=TRUE) # Not so recommended
par(mfrow=c(2,1))
plot(fit1, se=TRUE, scol="blue")
fit2 = vglm(yy ~ bs(x), gpd(threshold), trace=TRUE) # More recommended
plotvgam(fit2, se=TRUE, scol="blue")
## End(Not run)
```

---

gpdUC *The Generalized Pareto Distribution*

Description

Density, distribution function, quantile function and random generation for the generalized Pareto distribution (GPD) with location parameter `location`, scale parameter `scale` and shape parameter `shape`.

Usage

```
dgpd(x, location=0, scale=1, shape=0)
pgpd(q, location=0, scale=1, shape=0)
qgpd(p, location=0, scale=1, shape=0)
rgpd(n, location=0, scale=1, shape=0)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. Positive integer of length 1.
`location`	the location parameter μ.
`scale`	the scale parameter σ.
`shape`	the shape parameter ξ.

Details

See gpd, the **VGAM** family function for estimating the two parameters by maximum likelihood estimation, for formulae and other details. Apart from n, all the above arguments may be vectors and are recyled to the appropriate length if necessary.

Value

dgpd gives the density, pgpd gives the distribution function, qgpd gives the quantile function, and rgpd generates random deviates.

Note

The default values of all three parameters, especially $\xi = 0$, means the default distribution is the exponential.

Currently, these functions have different argument names compared with those in the **evd** package.

Author(s)

T. W. Yee

References

Coles, S. (2001) *An Introduction to Statistical Modeling of Extreme Values*. London: Springer-Verlag.

See Also

gpd.

Examples

```
## Not run:
x = seq(-0.2, 3, by=0.01)
loc = 0; sigma = 1; xi = -0.4
plot(x, dgpd(x, loc, sigma, xi), type="l", col="blue", ylim=c(0,1),
     main="Blue is density, red is cumulative distribution function",
     sub="Purple are 5,10,...,95 percentiles", ylab="", las=1)
abline(h=0, col="blue", lty=2)
lines(qgpd(seq(0.05,0.95,by=0.05), loc, sigma, xi),
      dgpd(qgpd(seq(0.05,0.95,by=0.05), loc, sigma, xi), loc, sigma, xi),
      col="purple", lty=3, type="h")
lines(x, pgpd(x, loc, sigma, xi), type="l", col="red")
abline(h=0, lty=2)
## End(Not run)
```

grc *Fitting Goodman's RC Association Model*

Description

Fits a Goodman's RC Association Model to a matrix of counts

Usage

```
grc(y, Rank = 1, Index.corner = 2:(1 + Rank),
    Structural.zero = 1, summary.arg = FALSE, h.step = 1e-04, ...)
```

Arguments

y	A matrix of counts. Output from `table()` is acceptable; it is converted into a matrix. Note that `y` must be at least 3 by 3.
Rank	An integer in the range 1,...,`min(nrow(y), ncol(y))`. This is the dimension of the fit.
Index.corner	A vector of `Rank` integers. These are used to store the `Rank` by `Rank` identity matrix in the `A` matrix; corner constraints are used.
Structural.zero	An integer in the range 1,...,`min(nrow(y), ncol(y))`, specifying the row that is used as the structural zero.
summary.arg	Logical. If `TRUE`, a summary is returned. If `TRUE`, `y` may be the output (fitted object) of `grc()`.
h.step	A small positive value that is passed into `summary.rrvglm()`. Only used when `summary.arg=TRUE`.
...	Arguments that are passed into `rrvglm.control()`.

Details

Goodman's RC association model can fit a reduced-rank approximation to a table of counts. The log of each cell mean is decomposed as an intercept plus a row effect plus a column effect plus a reduced-rank part. The latter can be collectively written `A %*% t(C)`, the product of two 'thin' matrices. Indeed, `A` and `C` have `Rank` columns. By default, the first column and row of the interaction matrix `A %*% t(C)` is chosen to be structural zeros, because `Structural.zero=1`. This means the first row of `A` are all zeros.

This function uses `options()$contrasts` to set up the row and column indicator variables.

Value

An object of class `"grc"`, which currently is the same as an `"rrvglm"` object.

Warning

This function temporarily creates a permanent data frame called `.grc.df`, which used to be needed by `summary.rrvglm()`. Then `.grc.df` is deleted before exiting the function. If an error occurs, then `.grc.df` may be present in the workspace.

Note

This function sets up variables etc. before calling `rrvglm()`. The `...` is passed into `rrvglm.control()`, meaning, e.g., `Rank=1` is default. Seting `trace=TRUE` may be useful for monitoring convergence.

Using `criterion="coefficients"` can result in slow convergence.

If `summary=TRUE`, then `y` can be a `"grc"` object, in which case a summary can be returned. That is, `grc(y, summary=TRUE)` is equivalent to `summary(grc(y))`.

Author(s)

Thomas W. Yee

References

Goodman, L. A. (1981) Association models and canonical correlation in the analysis of cross-classifications having ordered categories. *Journal of the American Statistical Association*, **76**, 320–334.

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information about the setting up of the indicator variables.

See Also

`rrvglm`, `rrvglm.control`, `rrvglm-class`, `summary.grc`, `auuc`.

Examples

```
# Some undergraduate student enrolments at the University of Auckland in 1990
data(auuc)
g1 = grc(auuc, Rank=1)
fitted(g1)
summary(g1)

g2 = grc(auuc, Rank=2, Index.corner=c(2,5))
fitted(g2)
summary(g2)
```

gumbel *Gumbel Distribution Family Function*

Description

Maximum likelihood estimation of the 2-parameter Gumbel distribution.

Usage

```
gumbel(llocation = "identity", lscale = "loge",
       elocation = list(), escale = list(),
       iscale = NULL, R = NA, percentiles = c(95, 99), mpv = FALSE,
       zero = NULL)
egumbel(llocation = "identity", lscale = "loge",
        elocation = list(), escale = list(),
        iscale = NULL, R = NA, percentiles = c(95, 99), mpv = FALSE,
        zero = NULL)
```

Arguments

`llocation, lscale`
: Parameter link functions for μ and σ. See `Links` for more choices.

`elocation, escale`
: Extra argument for the `llocation` and `lscale` arguments. See `earg` in `Links` for general information.

`iscale`
: Numeric and positive. Optional initial value for σ. Recycled to the appropriate length. In general, a larger value is better than a smaller value. A `NULL` means an initial value is computed internally.

`R`
: Numeric. Maximum number of values possible. See **Details** for more details.

`percentiles`
: Numeric vector of percentiles used for the fitted values. Values should be between 0 and 100. This argument uses the argument `R` if assigned. If `percentiles=NULL` then the mean will be returned as the fitted values.

`mpv`
: Logical. If `mpv=TRUE` then the *median predicted value* (MPV) is computed and returned as the (last) column of the fitted values. This argument is ignored if `percentiles=NULL`. See **Details** for more details.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The value (possibly values) must be from the set {1,2} corresponding respectively to μ and σ. By default all linear/additive predictors are modelled as a linear combination of the explanatory variables.

Details

The Gumbel distribution is a generalized extreme value (GEV) distribution with *shape* parameter $\xi = 0$. Consequently it is more easily estimated than the GEV. See `gev` for more details.

The quantity R is the maximum number of observations possible, for example, in the Venice data below, the top 10 daily values are recorded for each year, therefore $R = 365$ because there are about

365 days per year. The MPV is the value of the response such that the probability of obtaining a value greater than the MPV is 0.5 out of R observations. For the Venice data, the MPV is the sea level such that there is an even chance that the highest level for a particular year exceeds the MPV. When `mpv=TRUE`, the column labelled `"MPV"` contains the MPVs when `fitted()` is applied to the fitted object.

The formula for the mean of a response Y is $\mu + \sigma \times Euler$ where $Euler$ is a constant that has value approximately equal to 0.5772. The formula for the percentiles are (if `R` is not given) $\mu - \sigma \times \log[-\log(P/100)]$ where P is the `percentile` argument value(s). If `R` is given then the percentiles are $\mu - \sigma \times \log[R(1 - P/100)]$.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Warning

When `R` is not given (the default) the fitted percentiles are that of the data, and not of the overall population. For example, in the example below, the 50 percentile is approximately the running median through the data, however, the data are the highest sea level measurements recorded each year (it therefore equates to the median predicted value or MPV).

Note

`egumbel()` only handles a univariate response, and is preferred to `gumbel()` because it is faster.

`gumbel()` can handle a multivariate response, i.e., a matrix with more than one column. Each row of the matrix is sorted into descending order. Missing values in the response are allowed but require `na.action=na.pass`. The response matrix needs to be padded with any missing values. With a multivariate response one has a matrix `y`, say, where `y[,2]` contains the second order statistics etc.

Author(s)

T. W. Yee

References

Yee, T. W. and Stephenson, A. G. (2007) Vector generalized linear and additive extreme value models. *Extremes*, **10**, 1–19.

Smith, R. L. (1986) Extreme value theory based on the *r* largest annual events. *Journal of Hydrology*, **86**, 27–43.

Rosen, O. and Cohen, A. (1996) Extreme percentile regression. In: Haerdle, W. and Schimek, M. G. (eds.), *Statistical Theory and Computational Aspects of Smoothing: Proceedings of the COMPSTAT '94 Satellite Meeting held in Semmering, Austria, 27–28 August 1994*, pp.200–214, Heidelberg: Physica-Verlag.

Coles, S. (2001) *An Introduction to Statistical Modeling of Extreme Values*. London: Springer-Verlag.

See Also

rgumbel, cgumbel, guplot, gev, egev, venice.

Examples

```
# Example 1: Simulated data
y = rgumbel(n=1000, loc = 100, scale=exp(1))
fit = vglm(y ~ 1, egumbel(perc=NULL), trace=TRUE)
coef(fit, matrix=TRUE)
Coef(fit)
fitted(fit)[1:4,]
mean(y)

# Example 2: Venice data
data(venice)
(fit = vglm(cbind(r1,r2,r3,r4,r5) ~ year, data=venice,
           gumbel(R=365, mpv=TRUE), trace=TRUE))
fitted(fit)[1:5,]
coef(fit, mat=TRUE)
vcov(summary(fit))
sqrt(diag(vcov(summary(fit))))   # Standard errors

# Example 3: Try a nonparametric fit ----------------------
# Use the entire data set, including missing values
y = as.matrix(venice[,paste("r",1:10,sep="")])
fit1 = vgam(y ~ s(year, df=3), gumbel(R=365, mpv=TRUE),
            data=venice, trace=TRUE, na.action=na.pass)
fit1@y[4:5,]  # NAs used to pad the matrix

## Not run:
# Plot the component functions
par(mfrow=c(2,1), mar=c(5,4,.2,1)+0.1, xpd=TRUE)
plot(fit1, se=TRUE, lcol="blue", scol="green", lty=1,
     lwd=2, slwd=2, slty="dashed")

# Quantile plot --- plots all the fitted values
par(mfrow=c(1,1), bty="l", mar=c(4,4,.2,3)+0.1, xpd=TRUE, las=1)
qtplot(fit1, mpv=TRUE, lcol=c(1,2,5), tcol=c(1,2,5), lwd=2,
       pcol="blue", tadj=0.1, ylab="Sea level (cm)")

# Plot the 99 percentile only
par(mfrow=c(1,1), mar=c(3,4,.2,1)+0.1, xpd=TRUE)
year = venice[["year"]]
matplot(year, y, ylab="Sea level (cm)", type="n")
matpoints(year, y, pch="*", col="blue")
lines(year, fitted(fit1)[,"99%"], lwd=2, col="red")

# Check the 99 percentiles with a smoothing spline.
# Nb. (1-0.99) * 365 = 3.65 is approx. 4, meaning the 4th order
# statistic is approximately the 99 percentile.
par(mfrow=c(1,1), mar=c(3,4,2,1)+0.1, xpd=TRUE, lwd=2)
plot(year, y[,4], ylab="Sea level (cm)", type="n",
```

```
     main="Red is 99 percentile, Green is a smoothing spline")
points(year, y[,4], pch="4", col="blue")
lines(year, fitted(fit1)[,"99%"], lty=1, col="red")
lines(smooth.spline(year, y[,4], df=4), col="darkgreen", lty=2)
## End(Not run)
```

`gumbelIbiv` *Gumbel's Type I Bivariate Distribution Family Function*

Description

Estimate the association parameter of Gumbel's Type I bivariate distribution using maximum likelihood estimation.

Usage

```
gumbelIbiv(lapar="identity", earg=list(), iapar=NULL, method.init=1)
```

Arguments

`lapar`	Link function applied to the association parameter α. See `Links` for more choices.
`earg`	List. Extra argument for the link. See `earg` in `Links` for general information.
`iapar`	Numeric. Optional initial value for α. By default, an initial value is chosen internally. If a convergence failure occurs try assigning a different value. Assigning a value will override the argument `method.init`.
`method.init`	An integer with value `1` or `2` which specifies the initialization method. If failure to converge occurs try the other value, or else specify a value for `ia`.

Details

The cumulative distribution function is

$$P(Y_1 \leq y_1, Y_2 \leq y_2) = e^{-y_1-y_2+\alpha y_1 y_2} + 1 - e^{-y_1} - e^{-y_2}$$

for real α. The support of the function is for $y_1 > 0$ and $y_2 > 0$. The marginal distributions are an exponential distribution with unit mean.

A variant of Newton-Raphson is used, which only seems to work for an intercept model. It is a very good idea to set `trace=TRUE`.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Note

The response must be a two-column matrix. Currently, the fitted value is a matrix with two columns and values equal to 1. This is because each marginal distribution corresponds to a exponential distribution with unit mean.

This **VGAM** family function should be used with caution.

Author(s)

T. W. Yee

References

Castillo, E., Hadi, A. S., Balakrishnan, N. Sarabia, J. S. (2005) *Extreme Value and Related Models with Applications in Engineering and Science*, Hoboken, N.J.: Wiley-Interscience.

See Also

morgenstern.

Examples

```
n = 1000
ymat = cbind(rexp(n), rexp(n))
## Not run: plot(ymat)
fit = vglm(ymat ~ 1, fam=gumbelIbiv, trace=TRUE)
fit = vglm(ymat ~ 1, fam=gumbelIbiv, trace=TRUE, crit="coef")
coef(fit, matrix=TRUE)
Coef(fit)
fitted(fit)[1:5,]
```

---

gumbelUC *The Gumbel Distribution*

Description

Density, distribution function, quantile function and random generation for the Gumbel distribution with location parameter `location` and scale parameter `scale`.

Usage

```
dgumbel(x, location=0, scale=1)
pgumbel(q, location=0, scale=1)
qgumbel(p, location=0, scale=1)
rgumbel(n, location=0, scale=1)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. Positive integer of length 1.
`location`	the location parameter μ. This is not the mean of the Gumbel distribution (see **Details** below).
`scale`	the scale parameter σ. This is not the standard deviation of the Gumbel distribution (see **Details** below).

Details

The Gumbel distribution is a special case of the *generalized extreme value* (GEV) distribution where the shape parameter $\xi = 0$. The latter has 3 parameters, so the Gumbel distribution has two. The Gumbel distribution function is

$$G(y) = \exp\left(-\exp\left[-\frac{y-\mu}{\sigma}\right]\right)$$

where $-\infty < y < \infty$, $-\infty < \mu < \infty$ and $\sigma > 0$. Its mean is

$$\mu - \sigma * \gamma$$

and its variance is

$$\sigma^2 * \pi^2/6$$

where γ is Euler's constant (which can be obtained as `-digamma(1)`).

See `gumbel`, the **VGAM** family function for estimating the two parameters by maximum likelihood estimation, for formulae and other details. Apart from `n`, all the above arguments may be vectors and are recyled to the appropriate length if necessary.

Value

`dgumbel` gives the density, `pgumbel` gives the distribution function, `qgumbel` gives the quantile function, and `rgumbel` generates random deviates.

Note

The **VGAM** family function `gumbel` can estimate the parameters of a Gumbel distribution using maximum likelihood estimation.

Author(s)

T. W. Yee

References

Coles, S. (2001) *An Introduction to Statistical Modeling of Extreme Values*. London: Springer-Verlag.

See Also

gumbel, egumbel, gev.

Examples

```
mu = 1
sigma = 2
y = rgumbel(n=100, loc=mu, scale=sigma)
mean(y)
mu - sigma * digamma(1)  # population mean
var(y)
sigma^2 * pi^2 / 6       # population variance

## Not run:
x = seq(-2.5, 3.5, by=0.01)
loc = 0; sigma = 1
plot(x, dgumbel(x, loc, sigma), type="l", col="blue", ylim=c(0,1),
     main="Blue is density, red is cumulative distribution function",
     sub="Purple are 5,10,...,95 percentiles", ylab="", las=1)
abline(h=0, col="blue", lty=2)
lines(qgumbel(seq(0.05,0.95,by=0.05), loc, sigma),
      dgumbel(qgumbel(seq(0.05,0.95,by=0.05), loc, sigma), loc, sigma),
      col="purple", lty=3, type="h")
lines(x, pgumbel(x, loc, sigma), type="l", col="red")
abline(h=0, lty=2)
## End(Not run)
```

---

guplot *Gumbel Plot*

Description

Produces a Gumbel plot, a diagnostic plot for checking whether the data appears to be from a Gumbel distribution.

Usage

```
guplot(object, ...)
guplot.default(y, main="Gumbel Plot",
    xlab="Reduced data", ylab="Observed data", type="p", ...)
guplot.vlm(object, ...)
```

Arguments

y	A numerical vector. NAs etc. are not allowed.
main	Character. Overall title for the plot.
xlab	Character. Title for the x axis.
ylab	Character. Title for the y axis.

type	Type of plot. The default means points are plotted.
object	An object that inherits class "vlm", usually of class vglm-class or vgam-class.
...	Graphical argument passed into plot. See par for an exhaustive list. The arguments xlim and ylim are particularly useful.

Details

If Y has a Gumbel distribution then plotting the sorted values y_i versus the *reduced values* r_i should appear linear. The reduced values are given by

$$r_i = -\log(-\log(p_i))$$

where p_i is the ith plotting position, taken here to be $(i - 0.5)/n$. Here, n is the number of observations. Curvature upwards/downwards may indicate a Frechet/Weibull distribution, respectively. Outliers may also be detected using this plot.

The function guplot is generic, and guplot.default and guplot.vlm are some methods functions for Gumbel plots.

Value

A list is returned invisibly with the following components.

x	The reduced data.
y	The sorted y data.

Note

The Gumbel distribution is a special case of the GEV distribution with shape parameter equal to zero.

Author(s)

T. W. Yee

References

Coles, S. (2001) *An Introduction to Statistical Modeling of Extreme Values*. London: Springer-Verlag.

See Also

gumbel, egumbel, gev.

Examples

```
## Not run:
guplot(rnorm(500), las=1) -> i
names(i)
## End(Not run)
```

`hspider` *Hunting Spider Data*

Description

Abundance of hunting spiders in a Dutch dune area.

Usage

```
data(hspider)
```

Format

A data frame with 28 observations (sites) on the following 18 variables.

WaterCon Log percentage of soil dry mass.

BareSand Log percentage cover of bare sand.

FallTwig Log percentage cover of fallen leaves and twigs.

CoveMoss Log percentage cover of the moss layer.

CoveHerb Log percentage cover of the herb layer.

ReflLux Reflection of the soil surface with cloudless sky.

Alopacce Abundance of *Alopecosa accentuata*.

Alopcune Abundance of *Alopecosa cuneata*.

Alopfabr Abundance of *Alopecosa fabrilis*.

Arctlute Abundance of *Arctosa lutetiana*.

Arctperi Abundance of *Arctosa perita*.

Auloalbi Abundance of *Aulonia albimana*.

Pardlugu Abundance of *Pardosa lugubris*.

Pardmont Abundance of *Pardosa monticola*.

Pardnigr Abundance of *Pardosa nigriceps*.

Pardpull Abundance of *Pardosa pullata*.

Trocterr Abundance of *Trochosa terricola*.

Zoraspin Abundance of *Zora spinimana*.

Details

The data, which originally came from Van der Aart and Smeek-Enserink (1975) consists of abundances (numbers trapped over a 60 week period) and 6 environmental variables. There were 28 sites.

This data set has been often used to illustrate ordination, e.g., using canonical correspondence analysis (CCA). In the example below, the data is used for constrained quadratic ordination (CQO; formerly called canonical Gaussian ordination or CGO), a numerically intensive method that has many superior qualities. See `cqo` for details.

References

Van der Aart, P. J. M. and Smeek-Enserink, N. (1975) Correlations between distributions of hunting spiders (Lycosidae, Ctenidae) and environmental characteristics in a dune area. *Netherlands Journal of Zoology*, **25**, 1–45.

Examples

```
data(hspider)
str(hspider)

## Not run:
# Fit a rank-1 Poisson CQO
set.seed(111)  # This leads to the global solution
hspider[,1:6]=scale(hspider[,1:6]) # Standardize the environmental variables
p1 = cqo(cbind(Alopacce, Alopcune, Alopfabr, Arctlute, Arctperi, Auloalbi,
               Pardlugu, Pardmont, Pardnigr, Pardpull, Trocterr, Zoraspin) ~
         WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + ReflLux,
         fam = poissonff, data = hspider, Crow1posit=FALSE)
nos = ncol(p1@y)
lvplot(p1, y=TRUE, lcol=1:nos, pch=1:nos, pcol=1:nos)
Coef(p1)
summary(p1)


# Fit a rank-1 binomial CAO
hsbin = hspider   # Binary species data
hsbin[,-(1:6)] = as.numeric(hsbin[,-(1:6)] > 0)
set.seed(123)
ahsb1 = cao(cbind(Alopcune,Arctlute,Auloalbi,Zoraspin) ~
            WaterCon + ReflLux, family = binomialff(mv=TRUE),
            df1.nl = 2.2, Bestof=3, data = hsbin)
par(mfrow=2:1, las=1)
lvplot(ahsb1, type="predictors", llwd=2, ylab="logit p", lcol=1:9)
persp(ahsb1, rug=TRUE, col=1:10, lwd=2)
coef(ahsb1)
## End(Not run)
```

`hunua` *Hunua Ranges Data*

Description

The `hunua` data frame has 392 rows and 18 columns. Altitude is explanatory, and there are binary responses (presence/absence = 1/0 respectively) for 17 plant species.

Usage

```
data(hunua)
```

Format

This data frame contains the following columns:

agaaus Agathis australis, or Kauri

beitaw Beilschmiedia tawa, or Tawa

corlae Corynocarpus laevigatus

cyadea Cyathea dealbata

cyamed Cyathea medullaris

daccup Dacrydium cupressinum

dacdac Dacrycarpus dacrydioides

eladen Elaecarpus dentatus

hedarb Hedycarya arborea

hohpop Species name unknown

kniexc Knightia excelsa, or Rewarewa

kuneri Kunzea ericoides

lepsco Leptospermum scoparium

metrob Metrosideros robusta

neslan Nestegis lanceolata

rhosap Rhopalostylis sapida

vitluc Vitex lucens, or Puriri

altitude meters above sea level

Details

These were collected from the Hunua Ranges, a small forest in southern Auckland, New Zealand. At 392 sites in the forest, the presence/absence of 17 plant species was recorded, as well as the altitude. Each site was of area size $200m^2$.

Source

Dr Neil Mitchell, University of Auckland.

See Also

waitakere.

Examples

```
# Fit a GAM using vgam() and compare it with the Waitakere Ranges one
data(hunua)
fit.h = vgam(agaaus ~ s(altitude, df=2), binomialff, hunua)
## Not run:
plot(fit.h, se=TRUE, lcol="red", scol="red",
     main="Red is Hunua, Blue is Waitakere")
```

```
## End(Not run)
predict(fit.h, hunua, type="response")[1:3]

data(waitakere)
fit.w = vgam(agaaus ~ s(altitude, df=2), binomialff, waitakere)
## Not run:
plot(fit.w, se=TRUE, lcol="blue", scol="blue", add=TRUE)
## End(Not run)
predict(fit.w, hunua, type="response")[1:3]        # Same as above?
```

`hyperg` *Hypergeometric Family Function*

Description

Family function for a hypergeometric distribution where either the number of white balls or the total number of white and black balls are unknown.

Usage

```
hyperg(N=NULL, D=NULL, lprob="logit", earg=list(), iprob=NULL)
```

Arguments

`N`	Total number of white and black balls in the urn. Must be a vector with positive values, and is recycled, if necessary, to the same length as the response. One of `N` and `D` must be specified.
`D`	Number of white balls in the urn. Must be a vector with positive values, and is recycled, if necessary, to the same length as the response. One of `N` and `D` must be specified.
`lprob`	Link function for the probabilities. See `Links` for more choices.
`earg`	List. Extra argument for the link. See `earg` in `Links` for general information.
`iprob`	Optional initial value for the probabilities. The default is to choose initial values internally.

Details

Consider the scenario from `Hypergeometric` where there are $N = m+n$ balls in an urn, where m are white and n are black. A simple random sample (i.e., *without* replacement) of k balls is taken. The response here is the sample *proportion* of white balls. In this document, `N` is $N = m + n$, `D` is m (for the number of "defectives", in quality control terminology, or equivalently, the number of marked individuals). The parameter to be estimated is the population proportion of white balls, viz. $prob = m/(m + n)$.

Depending on which one of `N` and `D` is inputted, the estimate of the other parameter can be obtained from the equation $prob = m/(m + n)$, or equivalently, `prob = D/N`. However, the log-factorials are computed using `lgamma` and both m and n are not restricted to being integer. Thus if an integer N is to be estimated, it will be necessary to evaluate the likelihood function at integer values about the estimate, i.e., at `trunc(Nhat)` and `ceiling(Nhat)` where `Nhat` is the (real) estimate of N.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, vgam, rrvglm, cqo, and cao.

Warning

No checking is done to ensure that certain values are within range, e.g., $k \leq N$.

Note

The response can be of one of three formats: a factor (first level taken as success), a vector of proportions of success, or a 2-column matrix (first column = successes) of counts. The argument weights in the modelling function can also be specified. In particular, for a general vector of proportions, you will need to specify weights because the number of trials is needed.

Author(s)

Thomas W. Yee

References

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

See Also

Hypergeometric, binomialff.

Examples

```
nn = 100
m = 5   # number of white balls in the population
k = rep(4, len=nn)   # sample sizes
n = 4   # number of black balls in the population
y  = rhyper(nn=nn, m=m, n=n, k=k)
yprop = y / k  # sample proportions

# N is unknown, D is known. Both models are equivalent:
fit = vglm(cbind(y,k-y) ~ 1, hyperg(D=m), trace=TRUE, crit="c")
fit = vglm(yprop ~ 1, hyperg(D=m), weight=k, trace=TRUE, crit="c")

# N is known, D is unknown. Both models are equivalent:
fit = vglm(cbind(y,k-y) ~ 1, hyperg(N=m+n), trace=TRUE, crit="l")
fit = vglm(yprop ~ 1, hyperg(N=m+n), weight=k, trace=TRUE, crit="l")

coef(fit, matrix=TRUE)
Coef(fit)  # Should be equal to the true population proportion
unique(m / (m+n))  # The true population proportion
fit@extra
fitted(fit)[1:4]
summary(fit)
```

`hypersecant` *Hyperbolic Secant Distribution Family Function*

Description

Estimation of the parameter of the hyperbolic secant distribution.

Usage

```
hypersecant(link.theta="elogit", earg=if(link.theta=="elogit")
    list(min=-pi/2, max=pi/2) else list(), init.theta=NULL)
hypersecant.1(link.theta="elogit", earg=if(link.theta=="elogit")
    list(min=-pi/2, max=pi/2) else list(), init.theta=NULL)
```

Arguments

`link.theta`	Parameter link function applied to the parameter θ. See `Links` for more choices.
`earg`	List. Extra argument for the link. See `earg` in `Links` for general information.
`init.theta`	Optional initial value for θ. If failure to converge occurs, try some other value. The default means an initial value is determined internally.

Details

The probability density function of the hyperbolic secant distribution is given by

$$f(y) = \exp(\theta y + \log(\cos(\theta)))/(2\cosh(\pi y/2)),$$

for parameter $-\pi/2 < \theta < \pi/2$ and all real y. The mean of Y is $\tan(\theta)$ (returned as the fitted values).

Another parameterization is used for `hypersecant.1()`. This uses

$$f(y) = (\cos(\theta)/\pi) \times y^{-0.5+\theta/\pi} \times (1-y)^{-0.5-\theta/\pi},$$

for parameter $-\pi/2 < \theta < \pi/2$ and $0 < y < 1$. Then the mean of Y is $0.5 + \theta/\pi$ (returned as the fitted values) and the variance is $(\pi^2 - 4\theta^2)/(8\pi^2)$.

For both parameterizations Newton-Raphson is same as Fisher scoring.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Author(s)

T. W. Yee

References

Jorgensen, B. (1997) *The Theory of Dispersion Models*. London: Chapman & Hall.

See Also

elogit.

Examples

```
x = rnorm(n <- 200)
y = rnorm(n)  # Not very good data!
fit = vglm(y ~ x, hypersecant, trace=TRUE, crit="c")
coef(fit, matrix=TRUE)
fit@misc$earg

# Not recommended
fit = vglm(y ~ x, hypersecant(link="identity"), trace=TRUE, crit="c")
coef(fit, matrix=TRUE)
fit@misc$earg
```

---

hzeta *Haight's Zeta Family Function*

Description

Estimating the parameter of Haight's Zeta function.

Usage

```
hzeta(link = "loglog", earg=list(), init.alpha = NULL)
```

Arguments

link	Parameter link function for the parameter. See Links for more choices. Here, a log-log link keeps the parameter greater than one, meaning the mean is finite.
earg	List. Extra argument for the link. See earg in Links for general information.
init.alpha	Optional initial value for the (positive) parameter. The default is to obtain an initial value internally. Use this argument if the default fails.

Details

The probability function is

$$f(y) = (2y-1)^{(-\alpha)} - (2y+1)^{(-\alpha)},$$

where the parameter $\alpha > 0$ and $y = 1, 2, \ldots$. The function dhzeta computes this probability function. The mean of Y, which is returned as fitted values, is $(1 - 2^{-\alpha})\zeta(\alpha)$ provided $\alpha > 1$, where ζ is Riemann's zeta function. The mean is a decreasing function of α. The mean is infinite if $\alpha \leq 1$, and the variance is infinite if $\alpha \leq 2$.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Author(s)

T. W. Yee

References

Page 470 of Johnson N. L., Kotz S. and Kemp, A. W. (1993) *Univariate Discrete Distributions*, 2nd edition, Volume 2, New York: Wiley.

See Also

Hzeta, zeta, zetaff, loglog.

Examples

```
alpha = exp(exp(0.5))  # The parameter
y = rhzeta(n=1000, alpha) # Generate some hzeta random variates
fit = vglm(y ~ 1, hzeta, trace = TRUE, crit="c")
coef(fit, matrix=TRUE)
Coef(fit)  # Useful for intercept-only models; should be same as alpha
fitted(fit)[1:4,]
```

Hzeta *Haight's Zeta Function*

Description

Density, distribution function, quantile function and random generation for Haight's Zeta function distribution with parameter alpha.

Usage

```
dhzeta(x, alpha)
phzeta(q, alpha)
qhzeta(p, alpha)
rhzeta(n, alpha)
```

Arguments

x, q	Vector of quantiles. For the density, it should be a vector with positive integer values in order for the probabilities to be positive.
p	vector of probabilities.
n	number of observations. A single positive integer.
alpha	The parameter value. Must contain positive values and is recycled to the length of x or p or q if necessary.

Details

The probability function is

$$f(x) = (2x-1)^{(-\alpha)} - (2x+1)^{(-\alpha)},$$

where $\alpha > 0$ and $x = 1, 2, \ldots$.

Value

`dhzeta` gives the density, `phzeta` gives the distribution function, `qhzeta` gives the quantile function, and `rhzeta` generates random deviates.

Note

Given some response data, the **VGAM** family function `hzeta` estimates the parameter `alpha`.

Author(s)

T. W. Yee

References

Page 470 of Johnson N. L., Kotz S. and Kemp, A. W. (1993) *Univariate Discrete Distributions*, 2nd edition, Volume 2, New York: Wiley.

See Also

`hzeta`, `zeta`, `zetaff`.

Examples

```
dhzeta(1:20, 0.5)
rhzeta(20, 0.5)

round(1000 * dhzeta(1:8, 2))
table(rhzeta(1000, 2))

## Not run:
alpha = 0.5; x = 1:10
plot(x, dhzeta(x, alpha=alpha), type="h", ylim=0:1,
     sub="alpha=0.5", las=1, col="blue", ylab="Probability",
     main="Haight's zeta: blue=density; red=distribution function")
lines(x+0.1, phzeta(x, alpha=alpha), col="red", lty=3, type="h")
## End(Not run)
```

iam *Index from Array to Matrix*

Description

Maps the elements of an array containing symmetric positive-definite matrices to a matrix with sufficient columns to hold them (called matrix-band format.)

Usage

```
iam(j, k, M, hbw = M, both = FALSE, diagonal = TRUE)
```

Arguments

j	An integer from the set {1:M} giving the row number of an element.
k	An integer from the set {1:M} giving the column number of an element.
M	The number of linear/additive predictors. This is the dimension of each positive-definite symmetric matrix.
hbw	Defunct.
both	Logical. Return both the row and column indices? See below for more details.
diagonal	Logical. Return the indices for the diagonal elements? If FALSE then only the strictly upper triangular part of the matrix elements are used.

Details

Suppose we have n symmetric positive-definite square matrices, each M by M, and these are stored in an array of dimension c(n,M,M). Then these can be more compactly represented by a matrix of dimension c(n,K) where K is an integer between M and M*(M+1)/2 inclusive. The mapping between these two representations is given by this function. It firstly enumerates by the diagonal elements, followed by the band immediately above the diagonal, then the band above that one, etc. The last element is (1,M). This function performs the mapping from elements (j,k) of symmetric positive-definite square matrices to the columns of another matrix representing such. This is called the *matrix-band* format and is used by the **VGAM** package.

Value

This function has a dual purpose depending on the value of both. If both=FALSE then the column number corresponding to the j-k element of the matrix is returned. If both=TRUE then j and k are ignored and a list with the following components are returned.

row.index	The row indices of the upper triangular part of the matrix (This may or may not include the diagonal elements, depending on the argument diagonal).
col.index	The column indices of the upper triangular part of the matrix (This may or may not include the diagonal elements, depending on the argument diagonal).

Note

This function is used in the `weight` slot of many **VGAM** family functions (see `vglmff-class`), especially those whose M is determined by the data, e.g., `dirichlet`, `multinomial`.

Author(s)

T. W. Yee

References

The website `http://www.stat.auckland.ac.nz/~yee` contains some additional information.

See Also

`vglmff-class`.

Examples

```
iam(1, 2, M=3) # The 4th column represents element (1,2) of a 3x3 matrix
iam(NULL, NULL, M=3, both=TRUE) # Return the row and column indices

dirichlet()@weight

M = 4
temp1 = iam(NA, NA, M=M, both=TRUE)
mat1 = matrix(NA, M, M)
mat1[cbind(temp1$row, temp1$col)] = 1:length(temp1$row)
mat1 # More commonly used

M = 4
temp2 = iam(NA, NA, M=M, both=TRUE, diagonal=FALSE)
mat2 = matrix(NA, M, M)
mat2[cbind(temp2$row, temp2$col)] = 1:length(temp2$row)
mat2 # Rarely used
```

---

identity *Identity Link Function*

Description

Computes the identity transformation, including its inverse and the first two derivatives.

Usage

```
identity(theta, earg = list(), inverse = FALSE, deriv = 0,
         short = TRUE, tag = FALSE)
nidentity(theta, earg = list(), inverse = FALSE, deriv = 0,
          short = TRUE, tag = FALSE)
```

Arguments

theta	Numeric or character. See below for further details.
earg	Extra argument for passing in additional information. Here, the argument is unused.
inverse	Logical. If TRUE the inverse function is computed.
deriv	Order of the derivative. Integer with value 0, 1 or 2.
short	Used for labelling the blurb slot of a vglmff-class object.
tag	Used for labelling the linear/additive predictor in the initialize slot of a vglmff-class object. Contains a little more information if TRUE.

Details

The identity link function $g(\theta) = \theta$ should be available to every parameter estimated by the **VGAM** library. However, it usually results in numerical problems because the estimates lie outside the permitted range. Consequently, the result may contain Inf, -Inf, NA or NaN. The arguments short and tag are used only if theta is character.

The function nidentity is the negative-identity link function and corresponds to $g(\theta) = -\theta$. This is useful for some models, e.g., in the literature supporting the egev function it seems that half of the authors use $\xi = -k$ for the shape parameter and the other half use k instead of ξ.

Value

For identity(): for deriv = 0, the identity of theta, i.e., theta when inverse = FALSE, and if inverse = TRUE then theta. For deriv = 1, then the function returns *d* theta / *d* eta as a function of theta if inverse = FALSE, else if inverse = TRUE then it returns the reciprocal.

For nidentity(): the results are similar to identity() except for a sign change in most cases.

Author(s)

Thomas W. Yee

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

See Also

Links, loge, logit, probit, powl.

Examples

```
identity((-5):5)
identity((-5):5, deriv=1)
identity((-5):5, deriv=2)
nidentity((-5):5)
nidentity((-5):5, deriv=1)
nidentity((-5):5, deriv=2)
```

---

inv.gaussianff *Inverse Gaussian Distribution Family Function*

Description

Estimates the two parameters of the inverse Gaussian distribution by maximum likelihood estimation.

Usage

```
inv.gaussianff(lmu="loge", llambda="loge",
               emu=list(), elambda=list(),
               ilambda=1, zero=NULL)
```

Arguments

lmu, llambda	Parameter link functions for the μ and λ parameters. See Links for more choices.
emu, elambda	List. Extra argument for each of the links. See earg in Links for general information.
ilambda	Initial value for the λ parameter.
zero	An integer-valued vector specifying which linear/additive predictors η_j are modelled as intercepts only. The values must be from the set {1,2}.

Details

The inverse Gaussian distribution has a density that can be written as

$$f(y;\mu,\lambda) = \sqrt{\lambda/(2\pi y^3)}\,\exp\left(-\lambda(y-\mu)^2/(2\mu^2 y)\right)$$

where $y > 0$, $\mu > 0$, and $\lambda > 0$. The mean of Y is μ and its variance is μ^3/λ. By default, $\eta_1 = \log(\mu)$ and $\eta_2 = \log(\lambda)$.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Note

The inverse Gaussian distribution can be fitted (to a certain extent) using the usual GLM framework involving a scale parameter. This family function is different from that approach in that it estimates both parameters by full maximum likelihood estimation.

Author(s)

T. W. Yee

References

Johnson, N. L. and Kotz, S. and Balakrishnan, N. (1994) *Continuous Univariate Distributions*, 2nd edition, Volume 1, New York: Wiley.

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

See Also

`Inv.gaussian`, `wald`, `bisa`.

The R package **SuppDists** has several functions for evaluating the density, distribution function, quantile function and generating random numbers from the inverse Gaussian distribution.

Examples

```
n = 1000
shape = exp(3)
y = rinv.gaussian(n=n, mu=exp(2), lambda=shape)
fit = vglm(y ~ 1, inv.gaussianff(ilam=shape), trace=TRUE)
coef(fit, matrix=TRUE)
Coef(fit)
summary(fit)
```

---

invlomax *Inverse Lomax Distribution Family Function*

Description

Maximum likelihood estimation of the 2-parameter inverse Lomax distribution.

Usage

```
invlomax(link.scale = "loge", link.p = "loge",
         earg.scale=list(), earg.p=list(),
         init.scale = NULL, init.p = 1, zero = NULL)
```

Arguments

`link.scale, link.p`
: Parameter link functions applied to the (positive) scale parameter `scale` and (positive) shape parameter `p`. See `Links` for more choices.

`earg.scale, earg.p`
: List. Extra argument for each of the links. See `earg` in `Links` for general information.

`init.scale, init.p`
: Optional initial values for `scale` and `p`.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. Here, the values must be from the set {1,2} which correspond to `scale`, `p`, respectively.

Details

The 2-parameter inverse Lomax distribution is the 4-parameter generalized beta II distribution with shape parameters $a = q = 1$. It is also the 3-parameter Dagum distribution with shape parameter $a = 1$, as well as the beta distribution of the second kind with $q = 1$. More details can be found in Kleiber and Kotz (2003).

The inverse Lomax distribution has density

$$f(y) = py^{p-1}/[b^p\{1 + y/b\}^{p+1}]$$

for $b > 0$, $p > 0$, $y > 0$. Here, b is the scale parameter `scale`, and `p` is a shape parameter. The mean does not seem to exist.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

If the self-starting initial values fail, try experimenting with the initial value arguments, especially those whose default value is not `NULL`.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

See Also

`Invlomax`, `genbetaII`, `betaII`, `dagum`, `sinmad`, `fisk`, `lomax`, `paralogistic`, `invparalogistic`.

Examples

```
y = rinvlomax(n=2000, 6, 2)
fit = vglm(y ~ 1, invlomax, trace=TRUE)
fit = vglm(y ~ 1, invlomax, trace=TRUE, crit="c")
coef(fit, mat=TRUE)
Coef(fit)
summary(fit)
```

`invparalogistic` *Inverse Paralogistic Distribution Family Function*

Description

Maximum likelihood estimation of the 2-parameter inverse paralogistic distribution.

Usage

```
invparalogistic(link.a = "loge", link.scale = "loge",
                earg.a=list(), earg.scale=list(),
                init.a = 1, init.scale = NULL, zero = NULL)
```

Arguments

`link.a, link.scale`
: Parameter link functions applied to the (positive) shape parameter `a` and (positive) scale parameter `scale`. See `Links` for more choices.

`earg.a, earg.scale`
: List. Extra argument for each of the links. See `earg` in `Links` for general information.

`init.a, init.scale`
: Optional initial values for `a` and `scale`.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. Here, the values must be from the set {1,2} which correspond to `a`, `scale`, respectively.

Details

The 2-parameter inverse paralogistic distribution is the 4-parameter generalized beta II distribution with shape parameter $q = 1$ and $a = p$. It is the 3-parameter Dagum distribution with $a = p$. More details can be found in Kleiber and Kotz (2003).

The inverse paralogistic distribution has density

$$f(y) = a^2 y^{a^2-1}/[b^{a^2}\{1 + (y/b)^a\}^{a+1}]$$

for $a > 0$, $b > 0$, $y > 0$. Here, b is the scale parameter `scale`, and a is the shape parameter. The mean is

$$E(Y) = b\,\Gamma(a + 1/a)\,\Gamma(1 - 1/a)/\Gamma(a)$$

provided $a > 1$.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

If the self-starting initial values fail, try experimenting with the initial value arguments, especially those whose default value is not NULL.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

See Also

Invparalogistic, genbetaII, betaII, dagum, sinmad, fisk, invlomax, lomax, paralogistic.

Examples

```
y = rinvparalogistic(n=3000, 4, 6)
fit = vglm(y ~ 1, invparalogistic, trace=TRUE)
fit = vglm(y ~ 1, invparalogistic(init.a=2.7, init.sc=3.3),
           trace=TRUE, crit="c")
coef(fit, mat=TRUE)
Coef(fit)
summary(fit)
```

---

is.smart *Test For a Smart Object*

Description

Tests an object to see if it is smart.

Usage

```
is.smart(object)
```

Arguments

object	a function or a fitted model.

Details

If `object` is a function then this function looks to see whether `object` has the logical attribute `"smart"`. If so then this is returned, else `FALSE`.

If `object` is a fitted model then this function looks to see whether `object@smart.prediction` or `object$smart.prediction` exists. If it does and it is not equal to `list(smart.arg=FALSE)` then a `TRUE` is returned, else `FALSE`. The reason for this is because, e.g., `lm(...,smart=FALSE)` and `vglm(...,smart=FALSE)`, will return such a specific list.

Writers of smart functions manually have to assign this attribute to their smart function after it has been written.

Value

Returns `TRUE` or `FALSE`, according to whether the `object` is smart or not.

Examples

```
is.smart(my1)  # TRUE
is.smart(poly) # TRUE
if(is.R()) library(splines)
is.smart(bs)   # TRUE
is.smart(ns)   # TRUE
is.smart(tan)  # FALSE
if(!is.R()) is.smart(lm)   # TRUE
## Not run:
library(VGAM)
x = rnorm(9)
fit1 = vglm(rnorm(9) ~ x, normal1)
is.smart(fit1)   # TRUE
fit2 = vglm(rnorm(9) ~ x, normal1, smart=FALSE)
is.smart(fit2)   # FALSE
fit2@smart.prediction
## End(Not run)
```

---

laplace *Laplace Distribution*

Description

Maximum likelihood estimation of the 2-parameter Laplace distribution.

Usage

```
laplace(llocation="identity", lscale="loge",
        elocation=list(), escale=list(),
        ilocation=NULL, iscale=NULL,
        method.init=1, zero=NULL)
```

Arguments

`llocation, lscale`
: Character. Parameter link functions for location parameter a and scale parameter b. See `Links` for more choices.

`elocation, escale`
: List. Extra argument for each of the links. See `earg` in `Links` for general information.

`ilocation, iscale`
: Optional initial values. If given, it must be numeric and values are recycled to the appropriate length. The default is to choose the value internally.

`method.init`
: Initialization method. Either the value 1 or 2.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The value (possibly values) must be from the set {1,2} corresponding respectively to a and b. By default all linear/additive predictors are modelled as a linear combination of the explanatory variables.

Details

The Laplace distribution is often known as the *double-exponential* distribution and, for modelling, has heavier tail than the normal distribution. The Laplace density function is

$$f(y) = \frac{1}{2b} \exp\left(-\frac{|y-a|}{b}\right)$$

where $-\infty < y < \infty$, $-\infty < a < \infty$ and $b > 0$. Its mean is a and its variance is $2b^2$.

For `y ~ 1` (where `y` is the response) the maximum likelihood estimate (MLE) for the location parameter is the sample median, and the MLE for b is `mean(abs(y-location))` (replace location by its MLE if unknown).

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Warning

This family function has not been fully tested. The MLE regularity conditions do not hold for this distribution, therefore misleading inferences may result, e.g., in the `summary` and `vcov` of the object.

Note

This family function uses Fisher scoring. Convergence may be slow for non-intercept-only models; half-stepping is frequently required.

Author(s)

T. W. Yee

References

Kotz, S., Kozubowski, T. J. and Podgorski, K. (2001) *The Laplace distribution and generalizations: a revisit with applications to communications, economics, engineering, and finance*, Boston: Birkhauser.

See Also

rlaplace.

Examples

```
y = rlaplace(n <- 100, loc=2, scale=exp(1))
fit = vglm(y  ~ 1, laplace, trace=TRUE, crit="l")
coef(fit, matrix=TRUE)
Coef(fit)
median(y)

x = runif(n <- 1001)
y = rlaplace(n, loc=2, scale=exp(-1+1*x))
fit = vglm(y  ~ x, laplace(iloc=0.2, meth=2, zero=1), trace=TRUE)
coef(fit, matrix=TRUE)
```

---

laplaceUC *The Laplace Distribution*

Description

Density, distribution function, quantile function and random generation for the Laplace distribution with location parameter `location` and scale parameter `scale`.

Usage

```
dlaplace(x, location=0, scale=1)
plaplace(q, location=0, scale=1)
qlaplace(p, location=0, scale=1)
rlaplace(n, location=0, scale=1)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. Positive integer of length 1.
`location`	the location parameter a, which is the mean.
`scale`	the scale parameter b. Must consist of positive values.

Details

The Laplace distribution is often known as the double-exponential distribution and, for modelling, has heavier tail than the normal distribution. The Laplace density function is

$$f(y) = \frac{1}{2b} \exp\left(-\frac{|y-a|}{b}\right)$$

where $-\infty < y < \infty$, $-\infty < a < \infty$ and $b > 0$. The mean is a and the variance is $2b^2$.

See `laplace`, the **VGAM** family function for estimating the two parameters by maximum likelihood estimation, for formulae and details. Apart from `n`, all the above arguments may be vectors and are recyled to the appropriate length if necessary.

Value

`dlaplace` gives the density, `plaplace` gives the distribution function, `qlaplace` gives the quantile function, and `rlaplace` generates random deviates.

Author(s)

T. W. Yee

References

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

See Also

`laplace`.

Examples

```
loc = 1; b = 2
y = rlaplace(n=100, loc=loc, scale=b)
mean(y)
loc      # population mean
var(y)
2 * b^2  # population variance

## Not run:
x = seq(-5, 5, by=0.01)
loc = 0; b = 1.5
plot(x, dlaplace(x, loc, b), type="l", col="blue", ylim=c(0,1),
     main="Blue is density, red is cumulative distribution function",
     sub="Purple are 5,10,...,95 percentiles", las=1, ylab="")
abline(h=0, col="blue", lty=2)
lines(qlaplace(seq(0.05,0.95,by=0.05), loc, b),
      dlaplace(qlaplace(seq(0.05,0.95,by=0.05), loc, b), loc, b),
      col="purple", lty=3, type="h")
lines(x, plaplace(x, loc, b), type="l", col="red")
abline(h=0, lty=2)
```

```
## End(Not run)

plaplace(qlaplace(seq(0.05,0.95,by=0.05), loc, b), loc, b)
```

`leipnik` *Leipnik Distribution Family Function*

Description

Estimates the two parameters of a (transformed) Leipnik distribution by maximum likelihood estimation.

Usage

```
leipnik(lmu = "logit", llambda = "loge",
        emu=list(), elambda=list(),
        imu = NULL, ilambda = NULL)
```

Arguments

`lmu, llambda`	Link function for the μ and λ parameters. See `Links` for more choices.
`imu, ilambda`	Numeric. Optional initial values for μ and λ.
`emu, elambda`	List. Extra argument for each of the links. See `earg` in `Links` for general information.

Details

The (transformed) Leipnik distribution has density function

$$f(y;\mu,\lambda) = \frac{\{y(1-y)\}^{-\frac{1}{2}}}{\text{Beta}(\frac{\lambda+1}{2},\frac{1}{2})}\left[1+\frac{(y-\mu)^2}{y(1-y)}\right]^{-\frac{\lambda}{2}}$$

where $0 < y < 1$ and $\lambda > -1$. The mean is μ (returned as the fitted values) and the variance is $1/\lambda$.

Jorgensen (1997) calls the above the **transformed** Leipnik distribution, and if $y = (x+1)/2$ and $\mu = (\theta+1)/2$, then the distribution of X as a function of x and θ is known as the the (untransformed) Leipnik distribution. Here, both x and θ are in $(-1, 1)$.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

Warning

If `llambda="identity"` then it is possible that the `lambda` estimate becomes less than -1, i.e., out of bounds. One way to stop this is to choose `llambda="loge"`, however, `lambda` is then constrained to be positive.

Note

Convergence may be slow or fail. Until better initial value estimates are forthcoming try assigning the argument `ilambda` some numerical value if it fails to converge. Currently, Newton-Raphson is implemented, not Fisher scoring. Currently, this family function probably only really works for intercept-only models, i.e., `y ~ 1` in the formula.

Author(s)

T. W. Yee

References

Jorgensen, B. (1997) *The Theory of Dispersion Models*. London: Chapman & Hall

Johnson, N. L. and Kotz, S. and Balakrishnan, N. (1995) *Continuous Univariate Distributions*, 2nd edition, Volume 2, New York: Wiley. (pages 612–617).

See Also

`mccullagh89`.

Examples

```
y = rnorm(n=2000, mean=0.5, sd=0.1)  # Not good data
fit = vglm(y ~ 1, leipnik(ilambda=1), tr=TRUE, checkwz=FALSE)
fit = vglm(y ~ 1, leipnik(ilambda=1,llam=logoff, elam=list(offset=1)),
           trace=TRUE, cri="coef")
fitted(fit)[1:5]
mean(y)
summary(fit)
coef(fit, matrix=TRUE)
Coef(fit)

sum(weights(fit))  # sum of the prior weights
sum(weights(fit, type="w")) # sum of the working weights
```

`lerch` *Lerch Phi Function*

Description

Computes the Lerch transcendental Phi function.

Usage

```
lerch(x, s, v, tolerance=1.0e-10, iter=100)
```

Arguments

`x, s, v`	Numeric. This function recyles values of `x`, `s`, and `v` if necessary.
`tolerance`	Numeric. Accuracy required, must be positive and less than 0.01.
`iter`	Maximum number of iterations allowed to obtain convergence. If `iter` is too small then a result of `NA` may occur; if so, try increasing its value.

Details

The Lerch transcendental function is defined by

$$\Phi(x, s, v) = \sum_{n=0}^{\infty} \frac{x^n}{(n+v)^s}$$

where $|x| < 1$ and $v \neq 0, -1, -2, \ldots$. Actually, x may be complex but this function only works for real x. The algorithm used is based on the relation

$$\Phi(x, s, v) = x^m \Phi(x, s, v+m) + \sum_{n=0}^{m-1} \frac{x^n}{(n+v)^s}.$$

See the URL below for more information. This function is a wrapper function for the C code described below.

Value

Returns the value of the function evaluated at the values of `x`, `s`, `v`. If the above ranges of x and v are not satisfied, or some numeric problems occur, then this function will return a `NA` for those values.

Warning

This function has not been thoroughly tested and contains bugs, for example, the zeta function cannot be computed with this function even though $\zeta(s) = \Phi(x = 1, s, v = 1)$. There are many sources of problems such as lack of convergence, overflow and underflow, especially near singularities. If any problems occur then a `NA` will be returned.

Note

There are a number of special cases, e.g., the Riemann zeta-function is given by $\zeta(s) = \Phi(x = 1, s, v = 1)$. The special case of $s = 1$ corresponds to the hypergeometric 2F1, and this is implemented in the **gsl** package. The Lerch transcendental Phi function should not be confused with the Lerch zeta function though they are quite similar.

Author(s)

S. V. Aksenov and U. D. Jentschura wrote the C code. The R wrapper function was written by T. W. Yee.

References

http://aksenov.freeshell.org/lerchphi/source/lerchphi.c.

Bateman, H. (1953) *Higher Transcendental Functions*. Volume 1. McGraw-Hill, NY, USA.

See Also

zeta.

Examples

```
## Not run:
x = seq(-1.1, 1.1, len=201)
s=2; v=1
plot(x, lerch(x, s=s, v=v), type="l", col="red", las=1,
     main=paste("lerch(x, s=",s,", v=",v,")",sep=""))
abline(v=0, h=1, lty="dashed")

s = rnorm(n=100)
max(abs(zeta(s)-lerch(x=1,s=s,v=1))) # This fails (a bug); should be 0
## End(Not run)
```

---

levy *Levy Distribution Family Function*

Description

Estimates the two parameters of the Levy distribution by maximum likelihood estimation.

Usage

```
levy(delta = NULL, link.gamma = "loge", earg=list(),
     idelta = NULL, igamma = NULL)
```

Arguments

delta	Location parameter. May be assigned a known value, otherwise it is estimated (the default).
link.gamma	Parameter link function for the (positive) γ parameter. See Links for more choices.
earg	List. Extra argument for the link. See earg in Links for general information.
idelta	Initial value for the δ parameter (if it is to be estimated). By default, an initial value is chosen internally.
igamma	Initial value for the γ parameter. By default, an initial value is chosen internally.

Details

The Levy distribution is one of three stable distributions whose density function has a tractable form. The formula for the density is

$$f(y;\gamma,\delta) = \sqrt{\frac{\gamma}{2\pi}} \exp\left(\frac{-\gamma}{2(y-\delta)}\right) / (y-\delta)^{3/2}$$

where $\delta < y < \infty$ and $\gamma > 0$. The mean does not exist.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

If δ is given, then only one parameter is estimated and the default is $\eta_1 = \log(\gamma)$. If δ is not given, then $\eta_2 = \delta$.

Author(s)

T. W. Yee

References

Nolan, J. P. (2005) *Stable Distributions: Models for Heavy Tailed Data*.

See Also

The Nolan article is at `http://academic2.american.edu/~jpnolan/stable/chap1.pdf`.

Documentation accompanying the **VGAM** package at `http://www.stat.auckland.ac.nz/~yee` contains further information and examples.

Examples

```
n = 1000
mygamma = 1          # log link ==> 0 is the answer
delta = 0
y = delta + mygamma / rnorm(n)^2 # This is Levy(mygamma, delta)

# Cf. Table 1.1 of Nolan for Levy(1,0)
sum(y > 1) / length(y)  # Should be 0.6827
sum(y > 2) / length(y)  # Should be 0.5205

fit = vglm(y ~ 1, levy(delta=delta), trace=TRUE) # 1 parameter
fit = vglm(y ~ 1, levy(idelta=delta, igamma=mygamma),
           trace=TRUE)    # 2 parameters
coef(fit, matrix=TRUE)
Coef(fit)
summary(fit)
weights(fit, type="w")[1:4,]
```

lgammaUC *The Log-Gamma Distribution*

Description

Density, distribution function, quantile function and random generation for the log-gamma distribution with location parameter `location`, scale parameter `scale` and shape parameter `k`.

Usage

```
dlgamma(x, location=0, scale=1, k=1)
plgamma(q, location=0, scale=1, k=1)
qlgamma(p, location=0, scale=1, k=1)
rlgamma(n, location=0, scale=1, k=1)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. Positive integer of length 1.
`location`	the location parameter a.
`scale`	the (positive) scale parameter b.
`k`	the (positive) shape parameter k.

Details

See `lgammaff`, the **VGAM** family function for estimating the one parameter standard log-gamma distribution by maximum likelihood estimation, for formulae and other details. Apart from `n`, all the above arguments may be vectors and are recyled to the appropriate length if necessary.

Value

`dlgamma` gives the density, `plgamma` gives the distribution function, `qlgamma` gives the quantile function, and `rlgamma` generates random deviates.

Note

The **VGAM** family function `lgamma3ff` is for the three parameter (nonstandard) log-gamma distribution.

Author(s)

T. W. Yee

References

Kotz, S. and Nadarajah, S. (2000) *Extreme Value Distributions: Theory and Applications*, pages 48–49, London: Imperial College Press.

See Also

lgammaff, prentice74.

Examples

```
## Not run:
x = seq(-3.2, 5, by=0.01)
loc = 1
Scale = 1.5
k = 1.4
plot(x, dlgamma(x, loc, Scale, k), type="l", col="blue", ylim=c(0,1),
     main="Blue is density, red is cumulative distribution function",
     sub="Purple are 5,10,...,95 percentiles", las=1, ylab="")
abline(h=0, col="blue", lty=2)
lines(qlgamma(seq(0.05,0.95,by=0.05), loc, Scale, k),
      dlgamma(qlgamma(seq(0.05,0.95,by=0.05), loc, Scale, k), loc, Scale, k),
      col="purple", lty=3, type="h")
lines(x, plgamma(x, loc, Scale, k), type="l", col="red")
abline(h=0, lty=2)
## End(Not run)
```

---

lgammaff *Log-gamma Distribution Family Function*

Description

Estimation of the parameter of the standard and nonstandard log-gamma distribution.

Usage

```
lgammaff(link = "loge", earg=list(), init.k = NULL)
lgamma3ff(llocation="identity", lscale="loge", lshape="loge",
          elocation=list(), escale=list(), eshape=list(),
          ilocation=NULL, iscale=NULL, ishape=1, zero=NULL)
```

Arguments

llocation	Parameter link function applied to the location parameter a. See Links for more choices.
lscale	Parameter link function applied to the positive scale parameter b. See Links for more choices.
link, lshape	Parameter link function applied to the positive shape parameter k. See Links for more choices.

`earg, elocation, escale, eshape`
: List. Extra argument for each of the links. See `earg` in `Links` for general information.

`init.k, ishape`
: Initial value for k. If given, it must be positive. If failure to converge occurs, try some other value. The default means an initial value is determined internally.

`ilocation, iscale`
: Initial value for a and b. The defaults mean an initial value is determined internally for each.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,3}. The default value means none are modelled as intercept-only terms.

Details

The probability density function of the standard log-gamma distribution is given by

$$f(y) = \exp[ky - \exp(y)]/\Gamma(k),$$

for parameter $k > 0$ and all real y. The mean of Y is `digamma(k)` (returned as the fitted values) and its variance is `trigamma(k)`.

For the non-standard log-gamma distribution, one replaces y by $(y - a)/b$, where a is the location parameter and b is the positive scale parameter. Then the density function is

$$f(y) = \exp[k(y - a)/b - \exp((y - a)/b)]/(b\Gamma(k)).$$

The mean and variance of Y are `a + b*digamma(k)` (returned as the fitted values) and `b^2 * trigamma(k)`, respectively.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

The standard log-gamma distribution can be viewed as a generalization of the standard type 1 extreme value density: when $k = 1$ the distribution of $-Y$ is the standard type 1 extreme value distribution.

The standard log-gamma distribution is fitted with `lgammaff` and the non-standard (3-parameter) log-gamma distribution is fitted with `lgamma3ff`.

Author(s)

T. W. Yee

References

Kotz, S. and Nadarajah, S. (2000) *Extreme Value Distributions: Theory and Applications*, pages 48–49, London: Imperial College Press.

Johnson, N. L. and Kotz, S. and Balakrishnan, N. (1995) *Continuous Univariate Distributions*, 2nd edition, Volume 2, p.89, New York: Wiley.

See Also

`rlgamma`, `ggamma`, `prentice74`, `lgamma`.

Examples

```
y = rlgamma(n <- 100, k=exp(1))
fit = vglm(y ~ 1, lgammaff, trace=TRUE, crit="c")
summary(fit)
coef(fit, matrix=TRUE)
Coef(fit)

# Another example
x = runif(n <- 5000)
loc = -1 + 2*x
Scale = exp(1+x)
y = rlgamma(n, loc=loc, scale=Scale, k=exp(0))
fit = vglm(y ~ x, lgamma3ff(zero=3), trace=TRUE, crit="c")
coef(fit, matrix=TRUE)
```

---

lino *Generalized Beta Distribution Family Function*

Description

Maximum likelihood estimation of the 3-parameter generalized beta distribution as proposed by Libby and Novick (1982).

Usage

```
lino(lshape1="loge", lshape2="loge", llambda="loge",
     eshape1=list(), eshape2=list(), elambda=list(),
     ishape1=NULL, ishape2=NULL, ilambda=1, zero=NULL)
```

Arguments

`lshape1, lshape2`
: Parameter link functions applied to the two (positive) shape parameters a and b. See `Links` for more choices.

`llambda`
: Parameter link function applied to the parameter λ. See `Links` for more choices.

`eshape1, eshape2, elambda`
: List. Extra argument for each of the links. See `earg` in `Links` for general information.

`ishape1, ishape2, ilambda`
: Initial values for the parameters. A `NULL` value means one is computed internally. The argument `ilambda` must be numeric, and the default corresponds to a standard beta distribution.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. Here, the values must be from the set {1,2,3} which correspond to a, b, λ, respectively.

Details

Proposed by Libby and Novick (1982), this distribution has density

$$f(y; a, b, \lambda) = \frac{\lambda^a y^{a-1}(1-y)^{b-1}}{B(a,b)\{1-(1-\lambda)y\}^{a+b}}$$

for $a > 0$, $b > 0$, $\lambda > 0$, $0 < y < 1$. Here B is the beta function (see `beta`). The mean is a complicated function involving the Gauss hypergeometric function. If X has a `lino` distribution with parameters `shape1`, `shape2`, `lambda`, then $Y = \lambda X/(1-(1-\lambda)X)$ has a standard beta distribution with parameters `shape1`, `shape2`.

Since $\log(\lambda) = 0$ corresponds to the standard beta distribution, a `summary` of the fitted model performs a t-test for whether the data belongs to a standard beta distribution (provided the `loge` link for λ is used; this is the default).

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

The fitted values, which is usually the mean, have not been implemented yet and consequently are `NA`s.

Although Fisher scoring is used, the working weight matrices are positive-definite only in a certain region of the parameter space. Problems with this indicate poor initial values or an ill-conditioned model or insufficient data etc.

This model is can be difficult to fit. A reasonably good value of `ilambda` seems to be needed so if the self-starting initial values fail, try experimenting with the initial value arguments. Experience suggests `ilambda` is better a little larger, rather than smaller, compared to the true value.

Author(s)

T. W. Yee

References

Libby, D. L. and Novick, M. R. (1982) Multivariate generalized beta distributions with applications to utility assessment. *Journal of Educational Statistics*, **7**, 271–294.

Gupta, A. K. and Nadarajah, S. (2004) *Handbook of Beta Distribution and Its Applications*, NY: Marcel Dekker, Inc.

See Also

Lino, genbetaII.

Examples

```
y = rbeta(n=1000, exp(0.5), exp(1)) # Standard beta distribution
fit = vglm(y ~ 1, lino, trace=TRUE)
coef(fit, mat=TRUE)
Coef(fit)
fitted(fit)[1:4]
summary(fit)

# Nonstandard beta distribution
y = rlino(n=1000, shape1=2, shape2=3, lambda=exp(1))
fit = vglm(y ~ 1, lino(lshape1=identity, lshape2=identity, ilambda=10))
coef(fit, mat=TRUE)
```

Lino *The Generalized Beta Distribution (Libby and Novick, 1982)*

Description

Density, distribution function, quantile function and random generation for the generalized beta distribution, as proposed by Libby and Novick (1982).

Usage

```
dlino(x, shape1, shape2, lambda=1)
plino(q, shape1, shape2, lambda=1)
qlino(p, shape1, shape2, lambda=1)
rlino(n, shape1, shape2, lambda=1)
```

Arguments

x, q	vector of quantiles.
p	vector of probabilities.
n	number of observations. Must be a positive integer of length 1.
shape1, shape2, lambda	see lino.

Details

See lino, the **VGAM** family function for estimating the parameters, for the formula of the probability density function and other details.

Value

dlino gives the density, plino gives the distribution function, qlino gives the quantile function, and rlino generates random deviates.

Author(s)

T. W. Yee

See Also

lino.

Examples

```
## Not run:
lambda = 0.4
shape1 = exp(1.3)
shape2 = exp(1.3)
x = seq(0.0, 1.0, len=101)
plot(x, dlino(x, shape1=shape1, shape2=shape2, lambda=lambda),
     type="l", col="blue", las=1, ylab="",
     main="Blue is density, red is cumulative distribution function",
     sub="Purple lines are the 10,20,...,90 percentiles")
abline(h=0, col="blue", lty=2)
lines(x, plino(x, shape1=shape1, shape2=shape2, l=lambda), col="red")
probs = seq(0.1, 0.9, by=0.1)
Q = qlino(probs, shape1=shape1, shape2=shape2, lambda=lambda)
lines(Q, dlino(Q, shape1=shape1, shape2=shape2, lambda=lambda),
      col="purple", lty=3, type="h")
plino(Q, shape1=shape1, shape2=shape2, l=lambda) - probs # Should be all 0
## End(Not run)
```

---

lirat *Low-iron Rat Teratology Data*

Description

Low-iron rat teratology data.

Usage

```
data(lirat)
```

Format

A data frame with 58 observations on the following 4 variables.

N Litter size.

R Number of dead fetuses.

hb Hemoglobin level.

grp Group number. Group 1 is the untreated (low-iron) group, group 2 received injections on day 7 or day 10 only, group 3 received injections on days 0 and 7, and group 4 received injections weekly.

Details

The following description comes from Moore and Tsiatis (1991). The data comes from the experimental setup from Shepard et al. (1980), which is typical of studies of the effects of chemical agents or dietary regimens on fetal development in laboratory rats.

Female rats were put in iron-deficient diets and divided into 4 groups. One group of controls was given weekly injections of iron supplement to bring their iron intake to normal levels, while another group was given only placebo injections. Two other groups were given fewer iron-supplement injections than the controls. The rats were made pregnant, sacrificed 3 weeks later, and the total number of fetuses and the number of dead fetuses in each litter were counted.

For each litter the number of dead fetuses may be considered to be Binomial(N, p) where N is the litter size and p is the probability of a fetus dying. The parameter p is expected to vary from litter to litter, therefore the total variance of the proportions will be greater than that predicted by a binomial model, even when the covariates for hemoglobin level and experimental group are accounted for.

Source

Moore, D. F. and Tsiatis, A. (1991) Robust Estimation of the Variance in Moment Methods for Extra-binomial and Extra-Poisson Variation. *Biometrics*, **47**, 383–401.

References

Shepard, T. H., Mackler, B. and Finch, C. A. (1980) Reproductive studies in the iron-deficient rat. *Teratology*, **22**, 329–334.

Examples

```
## Not run:
data(lirat)
attach(lirat)
# cf. Figure 3 of Moore and Tsiatis (1991)
plot(hb, R/N, pch=as.character(grp), col=grp, las=1,
     xlab="Hemoglobin level", ylab="Proportion Dead")
detach(lirat)
## End(Not run)
```

lms.bcg *LMS Quantile Regression with a Box-Cox transformation to a Gamma Distribution*

Description

LMS quantile regression with the Box-Cox transformation to the gamma distribution.

Usage

```
lms.bcg(percentiles = c(25, 50, 75), zero = NULL,
        link.mu="identity", link.sigma = "loge",
        emu=list(), esigma=list(),
        dfmu.init=4, dfsigma.init=2,
        init.lambda = 1, init.sigma = NULL)
```

Arguments

In the following, n is the number of (independent) observations.

percentiles	A numerical vector containing values between 0 and 100, which are the quantiles. They will be returned as 'fitted values'.
zero	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,3}. The default value, NULL, means they all are functions of the covariates.
link.mu	Parameter link function applied to the second linear/additive predictor.
link.sigma	Parameter link function applied to the third linear/additive predictor. See Links for more choices.
emu, esigma	List. Extra argument for each of the links. See earg in Links for general information.
dfmu.init	Degrees of freedom for the cubic smoothing spline fit applied to get an initial estimate of mu. See vsmooth.spline.
dfsigma.init	Degrees of freedom for the cubic smoothing spline fit applied to get an initial estimate of sigma. See vsmooth.spline. This argument may be assigned NULL to get an initial value using some other algorithm.
init.lambda	Initial value for lambda. If necessary, it is recycled to be a vector of length n.
init.sigma	Optional initial value for sigma. If necessary, it is recycled to be a vector of length n. The default value, NULL, means an initial value is computed in the @initialize slot of the family function.

Details

Given a value of the covariate, this function applies a Box-Cox transformation to the response to best obtain a gamma distribution. The parameters chosen to do this are estimated by maximum likelihood or penalized maximum likelihood.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Warning

The computations are not simple, therefore convergence may fail. In that case, try different starting values. Also, the estimate may diverge quickly near the solution, in which case try prematurely stopping the iterations by assigning maxits to be the iteration number corresponding to the highest likelihood value. See the example below.

The expected value of the second derivative with respect to lambda may be incorrect (my calculations do not agree with the Lopatatzidis and Green manuscript.)

Note

The response must be positive because the Box-Cox transformation cannot handle negative values. The LMS-Yeo-Johnson-normal method can handle both positive and negative values.

In general, the lambda and sigma functions should be more smoother than the mean function. Often setting zero=1 or zero=3 or zero=c(1,3) is a good idea. See the example below.

While it is usual to regress the response against a single covariate, it is possible to add other explanatory variables, e.g., sex. See http://www.stat.auckland.ac.nz/~yee for further information and examples about this feature.

Author(s)

Thomas W. Yee

References

Lopatatzidis A. and Green, P. J. (unpublished manuscript) Semiparametric quantile regression using the gamma distribution.

Yee, T. W. (2004) Quantile regression via vector generalized additive models. *Statistics in Medicine*, **23**, 2295–2315.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

lms.bcn, lms.yjn, qtplot.lmscreg, deplot.lmscreg, cdf.lmscreg, bminz, alsqreg.

Examples

```
data(bminz)

# This converges, but can't be deplot()'ed or qtplot()'ed
fit = vglm(BMI ~ bs(age, df=4), fam=lms.bcg(zero=c(1,3)), data=bminz, tr=TRUE)
coef(fit, matrix=TRUE)
## Not run:
```

```
par(mfrow=c(1,1))
plotvgam(fit, se=TRUE) # Plot mu function (only)
## End(Not run)

# Difficult to get a model that converges
# Here, we prematurely stop iterations because it fails near the solution
fit = vgam(BMI ~ s(age, df=c(4,2)), maxit=4,
           fam=lms.bcg(zero=1, init.lam=3), data=bminz, tr=TRUE)
summary(fit)
predict(fit)[1:3,]
fitted(fit)[1:3,]
bminz[1:3,]
# Person 1 is near the lower quartile of BMI amongst people his age
cdf(fit)[1:3]

## Not run:
# Quantile plot
par(bty="l", mar=c(5,4,4,3)+0.1, xpd=TRUE)
qtplot(fit, percentiles=c(5,50,90,99), main="Quantiles",
       xlim=c(15,90), las=1, ylab="BMI", lwd=2, lcol=4)

# Density plot
ygrid = seq(15, 43, len=100)  # BMI ranges
par(mfrow=c(1,1), lwd=2)
a = deplot(fit, x0=20, y=ygrid, xlab="BMI", col="black",
    main="Density functions at Age = 20 (black), 42 (red) and 55 (blue)")
a
a = deplot(fit, x0=42, y=ygrid, add=TRUE, llty=2, col="red")
a = deplot(fit, x0=55, y=ygrid, add=TRUE, llty=4, col="blue", Attach=TRUE)
a@post$deplot  # Contains density function values
## End(Not run)
```

`lms.bcn` *LMS Quantile Regression with a Box-Cox Transformation to Normality*

Description

LMS quantile regression with the Box-Cox transformation to normality.

Usage

```
lms.bcn(percentiles = c(25, 50, 75), zero = NULL,
        link.mu="identity", link.sigma = "loge",
        emu=list(), esigma=list(),
        dfmu.init=4, dfsigma.init=2,
        init.lambda = 1, init.sigma = NULL)
```

Arguments

In the following, n is the number of (independent) observations.

percentiles	A numerical vector containing values between 0 and 100, which are the quantiles. They will be returned as 'fitted values'.
zero	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,3}. The default value, NULL, means they all are functions of the covariates.
link.mu	Parameter link function applied to the second linear/additive predictor. See Links for more choices.
link.sigma	Parameter link function applied to the third linear/additive predictor. See Links for more choices.
emu, esigma	List. Extra argument for each of the links. See earg in Links for general information.
dfmu.init	Degrees of freedom for the cubic smoothing spline fit applied to get an initial estimate of mu. See vsmooth.spline.
dfsigma.init	Degrees of freedom for the cubic smoothing spline fit applied to get an initial estimate of sigma. See vsmooth.spline. This argument may be assigned NULL to get an initial value using some other algorithm.
init.lambda	Initial value for lambda. If necessary, it is recycled to be a vector of length n.
init.sigma	Optional initial value for sigma. If necessary, it is recycled to be a vector of length n. The default value, NULL, means an initial value is computed in the @initialize slot of the family function.

Details

Given a value of the covariate, this function applies a Box-Cox transformation to the response to best obtain normality. The parameters chosen to do this are estimated by maximum likelihood or penalized maximum likelihood.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Warning

The computations are not simple, therefore convergence may fail. In that case, try different starting values. Also, the estimate may diverge quickly near the solution, in which case try prematurely stopping the iterations by assigning maxits to be the iteration number corresponding to the highest likelihood value.

Note

The response must be positive because the Box-Cox transformation cannot handle negative values. The LMS-Yeo-Johnson-normal method can handle both positive and negative values.

In general, the lambda and sigma functions should be more smoother than the mean function. Often setting `zero=1` or `zero=3` or `zero=c(1,3)` is a good idea. See the example below.

While it is usual to regress the response against a single covariate, it is possible to add other explanatory variables, e.g., sex. See http://www.stat.auckland.ac.nz/~yee for further information and examples about this feature.

Author(s)

Thomas W. Yee

References

Cole, T. J. and Green, P. J. (1992) Smoothing Reference Centile Curves: The LMS Method and Penalized Likelihood. *Statistics in Medicine*, **11**, 1305–1319.

Yee, T. W. (2004) Quantile regression via vector generalized additive models. *Statistics in Medicine*, **23**, 2295–2315.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

`lms.bcg`, `lms.yjn`, `qtplot.lmscreg`, `deplot.lmscreg`, `cdf.lmscreg`, `bminz`, `alsqreg`.

Examples

```
data(bminz)
fit = vgam(BMI ~ s(age, df=c(4,2)), fam=lms.bcn(zero=1), data=bminz, tr=TRUE)
predict(fit)[1:3,]
fitted(fit)[1:3,]
bminz[1:3,]
# Person 1 is near the lower quartile of BMI amongst people his age
cdf(fit)[1:3]

## Not run:
# Quantile plot
par(bty="l", mar=c(5,4,4,3)+0.1, xpd=TRUE)
qtplot(fit, percentiles=c(5,50,90,99), main="Quantiles",
       xlim=c(15,90), las=1, ylab="BMI", lwd=2, lcol=4)

# Density plot
ygrid = seq(15, 43, len=100)  # BMI ranges
par(mfrow=c(1,1), lwd=2)
a = deplot(fit, x0=20, y=ygrid, xlab="BMI", col="black",
    main="Density functions at Age = 20 (black), 42 (red) and 55 (blue)")
a
a = deplot(fit, x0=42, y=ygrid, add=TRUE, llty=2, col="red")
a = deplot(fit, x0=55, y=ygrid, add=TRUE, llty=4, col="blue", Attach=TRUE)
a@post$deplot  # Contains density function values
## End(Not run)
```

lms.yjn *LMS Quantile Regression with a Yeo-Johnson Transformation to Normality*

Description

LMS quantile regression with the Yeo-Johnson transformation to normality.

Usage

```
lms.yjn(percentiles = c(25, 50, 75), zero = NULL,
        link.lambda = "identity", link.sigma = "loge",
        elambda=list(), esigma=list(),
        dfmu.init=4, dfsigma.init=2,
        init.lambda = 1, init.sigma = NULL,
        rule = c(10, 5), yoffset = NULL,
        diagW=FALSE, iters.diagW=6)
```

Arguments

In the following, n is the number of (independent) observations.

percentiles	A numerical vector containing values between 0 and 100, which are the quantiles. They will be returned as 'fitted values'.
zero	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,3}. The default value, NULL, means they all are functions of the covariates.
link.lambda	Parameter link function applied to the first linear/additive predictor. See Links for more choices.
link.sigma	Parameter link function applied to the third linear/additive predictor. See Links for more choices.
elambda, esigma	List. Extra argument for each of the links. See earg in Links for general information.
dfmu.init	Degrees of freedom for the cubic smoothing spline fit applied to get an initial estimate of mu. See vsmooth.spline.
dfsigma.init	Degrees of freedom for the cubic smoothing spline fit applied to get an initial estimate of sigma. See vsmooth.spline. This argument may be assigned NULL to get an initial value using some other algorithm.
init.lambda	Initial value for lambda. If necessary, it is recycled to be a vector of length n.
init.sigma	Optional initial value for sigma. If necessary, it is recycled to be a vector of length n. The default value, NULL, means an initial value is computed in the @initialize slot of the family function.

rule	Number of abscissae used in the Gaussian integration scheme to work out elements of the weight matrices. The values given are the possible choices, with the first value being the default. The larger the value, the more accurate the approximation is likely to be but involving more computational expense.
yoffset	A value to be added to the response y, for the purpose of centering the response before fitting the model to the data. The default value, NULL, means -median(y) is used, so that the response actually used has median zero. The yoffset is saved on the object and used during prediction.
diagW	Logical. This argument is offered because the expected information matrix may not be positive-definite. Using the diagonal elements of this matrix results in a higher chance of it being positive-definite, however convergence will be very slow. If TRUE, then the first iters.diagW iterations will use the diagonal of the expected information matrix. The default is FALSE, meaning faster convergence.
iters.diagW	Integer. Number of iterations in which the diagonal elements of the expected information matrix are used. Only used if diagW = TRUE.

Details

Given a value of the covariate, this function applies a Yeo-Johnson transformation to the response to best obtain normality. The parameters chosen to do this are estimated by maximum likelihood or penalized maximum likelihood.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Warning

The computations are not simple, therefore convergence may fail. In that case, try different starting values.

The generic function predict, when applied to a lms.yjn fit, does not add back the yoffset value.

Note

The response may contain both positive and negative values. In contrast, the LMS-Box-Cox-normal and LMS-Box-Cox-gamma methods only handle a positive response because the Box-Cox transformation cannot handle negative values.

In general, the lambda and sigma functions should be more smoother than the mean function. Often setting zero=1 or zero=3 or zero=c(1,3) is a good idea. See the example below.

While it is usual to regress the response against a single covariate, it is possible to add other explanatory variables, e.g., sex. See http://www.stat.auckland.ac.nz/~yee for further information and examples about this feature.

Author(s)

Thomas W. Yee

References

Yeo, I.-K. and Johnson, R. A. (2000) A new family of power transformations to improve normality or symmetry. *Biometrika*, **87**, 954–959.

Yee, T. W. (2004) Quantile regression via vector generalized additive models. *Statistics in Medicine*, **23**, 2295–2315.

Yee, T. W. (2002) An Implementation for Regression Quantile Estimation. Pages 3–14. In: Haerdle, W. and Ronz, B., *Proceedings in Computational Statistics COMPSTAT 2002*. Heidelberg: Physica-Verlag.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

lms.bcn, lms.bcg, qtplot.lmscreg, deplot.lmscreg, cdf.lmscreg, bminz, alsqreg.

Examples

```
data(bminz)
fit = vgam(BMI ~ s(age, df=4), fam=lms.yjn(zero=c(1,3)),
           data=bminz, trace=TRUE)
predict(fit)[1:3,]
fitted(fit)[1:3,]
bminz[1:3,]
# Person 1 is near the lower quartile of BMI amongst people his age
cdf(fit)[1:3]

## Not run:
# Quantile plot
par(bty="l", mar=c(5,4,4,3)+0.1, xpd=TRUE)
qtplot(fit, percentiles=c(5,50,90,99), main="Quantiles",
       xlim=c(15,90), las=1, ylab="BMI", lwd=2, lcol=4)

# Density plot
ygrid = seq(15, 43, len=100)  # BMI ranges
par(mfrow=c(1,1), lwd=2)
a = deplot(fit, x0=20, y=ygrid, xlab="BMI", col="black",
    main="Density functions at Age = 20 (black), 42 (red) and 55 (blue)")
a
a = deplot(fit, x0=42, y=ygrid, add=TRUE, llty=2, col="red")
a = deplot(fit, x0=55, y=ygrid, add=TRUE, llty=4, col="blue", Attach=TRUE)
a@post$deplot  # Contains density function values
## End(Not run)
```

`Log` *Logarithmic Distribution*

Description

Density, distribution function, and random generation for the logarithmic distribution.

Usage

```
dlog(x, prob)
plog(q, prob, log.p=FALSE)
rlog(n, prob, Smallno=1.0e-6)
```

Arguments

`x, q`	Vector of quantiles. For the density, it should be a vector with positive integer values in order for the probabilities to be positive.
`n`	number of observations. A single positive integer.
`prob`	The parameter value c described in in `logff`. Here it is called `prob` because $0 < c < 1$ is the range. For `rlog()` this parameter must be of length 1.
`log.p`	Logical. If `TRUE` then all probabilities `p` are given as `log(p)`.
`Smallno`	Numeric, a small value used by the rejection method for determining the upper limit of the distribution. That is, `plog(U, prob) > 1-Smallno` where `U` is the upper limit.

Details

The details are given in `logff`.

Value

`dlog` gives the density, `plog` gives the distribution function, and `rlog` generates random deviates.

Note

Given some response data, the **VGAM** family function `logff` estimates the parameter `prob`.

Author(s)

T. W. Yee

References

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

See Also

logff.

Examples

```
dlog(1:20, 0.5)
rlog(20, 0.5)

## Not run:
prob = 0.8; x = 1:10
plot(x, dlog(x, prob=prob), type="h", ylim=0:1,
     sub="prob=0.8", las=1, col="blue", ylab="Probability",
     main="Logarithmic distribution: blue=density; red=distribution function")
lines(x+0.1, plog(x, prob=prob), col="red", lty=3, type="h")
## End(Not run)
```

---

logc *Complementary-log Link Function*

Description

Computes the complentary-log transformation, including its inverse and the first two derivatives.

Usage

```
logc(theta, earg = list(), inverse = FALSE, deriv = 0,
     short = TRUE, tag = FALSE)
```

Arguments

theta	Numeric or character. See below for further details.
earg	Optional list. Extra argument for passing in additional information. Values of theta which are less than or equal to 1 can be replaced by the bvalue component of the list earg before computing the link function value. The component name bvalue stands for "boundary value". See Links for general information about earg.
inverse	Logical. If TRUE the inverse function is computed.
deriv	Order of the derivative. Integer with value 0, 1 or 2.
short	Used for labelling the blurb slot of a vglmff-class object.
tag	Used for labelling the linear/additive predictor in the initialize slot of a vglmff-class object. Contains a little more information if TRUE.

Details

The complementary-log link function is suitable for parameters that are less than unity. Numerical values of theta close to 1 or out of range result in Inf, -Inf, NA or NaN. The arguments short and tag are used only if theta is character.

Value

For `deriv = 0`, the log of `theta`, i.e., `log(1-theta)` when `inverse = FALSE`, and if `inverse = TRUE` then `1-exp(theta)`.

For `deriv = 1`, then the function returns *d* `theta` / *d* `eta` as a function of `theta` if `inverse = FALSE`, else if `inverse = TRUE` then it returns the reciprocal.

Here, all logarithms are natural logarithms, i.e., to base *e*.

Note

Numerical instability may occur when `theta` is close to 1. One way of overcoming this is to use `earg`.

Author(s)

Thomas W. Yee

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

See Also

`Links`, `loge`, `cloglog`, `loglog`, `logoff`.

Examples

```
## Not run:
logc(seq(-0.2, 1.1, by=0.1))  # Has NAs
## End(Not run)
logc(seq(-0.2, 1.1, by=0.1), earg=list(bval=1-.Machine$double.eps))  # Has no NAs
```

---

loge *Log link function*

Description

Computes the log transformation, including its inverse and the first two derivatives.

Usage

```
loge(theta, earg = list(), inverse = FALSE, deriv = 0,
     short = TRUE, tag = FALSE)
nloge(theta, earg = list(), inverse = FALSE, deriv = 0,
      short = TRUE, tag = FALSE)
```

Arguments

theta	Numeric or character. See below for further details.
earg	Optional list. Extra argument for passing in additional information. Values of theta which are less than or equal to 0 can be replaced by the bvalue component of the list earg before computing the link function value. The component name bvalue stands for "boundary value". See Links for general information about earg.
inverse	Logical. If TRUE the inverse function is computed.
deriv	Order of the derivative. Integer with value 0, 1 or 2.
short	Used for labelling the blurb slot of a vglmff-class object.
tag	Used for labelling the linear/additive predictor in the initialize slot of a vglmff-class object. Contains a little more information if TRUE.

Details

The log link function is very commonly used for parameters that are positive. Numerical values of theta close to 0 or out of range result in Inf, -Inf, NA or NaN. The arguments short and tag are used only if theta is character.

The function loge computes $\log(\theta)$ whereas nloge computes $-\log(\theta) = \log(1/\theta)$.

Value

The following concerns loge. For deriv = 0, the log of theta, i.e., log(theta) when inverse = FALSE, and if inverse = TRUE then exp(theta). For deriv = 1, then the function returns d theta / d eta as a function of theta if inverse = FALSE, else if inverse = TRUE then it returns the reciprocal.

Here, all logarithms are natural logarithms, i.e., to base e.

Note

This function is called loge to avoid conflict with the log function.

Numerical instability may occur when theta is close to 0 unless earg is used.

Author(s)

Thomas W. Yee

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

See Also

Links, logit, logc, loglog, log, logoff.

Examples

```
## Not run:
loge(seq(-0.2, 0.5, by=0.1))
loge(seq(-0.2, 0.5, by=0.1), earg=list(bvalue= .Machine$double.xmin))
nloge(seq(-0.2, 0.5, by=0.1))
nloge(seq(-0.2, 0.5, by=0.1), earg=list(bvalue= .Machine$double.xmin))
## End(Not run)
```

---

logff *Logarithmic Distribution*

Description

Estimating the parameter of the logarithmic distribution.

Usage

```
logff(link = "logit", earg=list(), init.c = NULL)
```

Arguments

link	Parameter link function applied to the parameter c, which lies between 0 and 1. See Links for more choices.
earg	List. Extra argument for the link. See earg in Links for general information.
init.c	Optional initial value for the c parameter. If given, it often pays to start with a larger value, e.g., 0.95. The default is to choose an initial value internally.

Details

The logarithmic distribution is based on the logarithmic series, and is scaled to a probability function. Its probability function is $f(y) = ac^y/y$, for $y = 1, 2, 3, \ldots$, where $0 < c < 1$, and $a = -1/\log(1-c)$. The mean is $ac/(1-c)$ (returned as the fitted values) and variance is $ac(1-ac)/(1-c)^2$.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

The function log computes the natural logarithm. In the **VGAM** library, a link function with option loge corresponds to this.

Author(s)

T. W. Yee

References

Chapter 7 of Johnson N. L., Kotz S. and Kemp, A. W. (1993) *Univariate Discrete Distributions*, 2nd edition, Volume 2, New York: Wiley.

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

rlog, log, loge, logoff.

Examples

```
y = rlog(n=1000, prob=logit(0.2, inverse=TRUE))
fit = vglm(y ~ 1, logff, trace=TRUE, crit="c")
coef(fit, matrix=TRUE)
Coef(fit)
## Not run:
hist(y, prob=TRUE, breaks=seq(0.5, max(y)+0.5, by=1))
x = seq(1, max(y), by=1)
lines(x, dlog(x, Coef(fit)[1]), col="red", type="h")
## End(Not run)
```

---

logistic *Logistic Distribution Family Function*

Description

Estimates the location and scale parameters of the logistic distribution by maximum likelihood estimation.

Usage

```
logistic1(llocation="identity", elocation=list(),
          scale.arg=1, method.init=1)
logistic2(llocation="identity", lscale="loge",
          elocation=list(), escale=list(),
          ilocation=NULL, iscale=NULL, method.init=1, zero=NULL)
```

Arguments

llocation	Link function applied to the location parameter l. See Links for more choices.
elocation, escale	List. Extra argument for each of the links. See earg in Links for general information.
scale.arg	Known positive scale parameter (called s below).

`lscale`	Parameter link function applied to the scale parameter s. See `Links` for more choices.
`ilocation`	Initial value for the location l parameter. By default, an initial value is chosen internally using `method.init`. Assigning a value will override the argument `method.init`.
`iscale`	Initial value for the scale s parameter. By default, an initial value is chosen internally using `method.init`. Assigning a value will override the argument `method.init`.
`method.init`	An integer with value `1` or `2` which specifies the initialization method. If failure to converge occurs try the other value.
`zero`	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The default is none of them. If used, choose one value from the set {1,2}.

Details

The two-parameter logistic distribution has a density that can be written as

$$f(y; l, s) = \frac{\exp[-(y-l)/s]}{s\,(1+\exp[-(y-l)/s])^2}$$

where $s > 0$ is the scale parameter, and l is the location parameter. The response $-\infty < y < \infty$. The mean of Y (which is the fitted value) is l and its variance is $\pi^2 s^2/3$.

`logistic1` estimates the location parameter only while `logistic2` estimates both parameters. By default, $\eta_1 = l$ and $\eta_2 = \log(s)$ for `logistic2`.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

Note

Fisher scoring is used, and the Fisher information matrix is diagonal.

Author(s)

T. W. Yee

References

Johnson, N. L. and Kotz, S. and Balakrishnan, N. (1994) *Continuous Univariate Distributions*, 2nd edition, Volume 1, New York: Wiley. Chapter 15.

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

Castillo, E., Hadi, A. S., Balakrishnan, N. Sarabia, J. S. (2005) *Extreme Value and Related Models with Applications in Engineering and Science*, Hoboken, N.J.: Wiley-Interscience, p.130.

deCani, J. S. and Stine, R. A. (1986) A note on Deriving the Information Matrix for a Logistic Distribution, *The American Statistician*, **40**, 220–222.

See Also

rlogis, bilogistic4.

Examples

```
# location unknown, scale known
n = 500
x = runif(n)
y = rlogis(n, loc=1+5*x, scale=4)
fit = vglm(y ~ x, logistic1(scale=4), trace=TRUE, crit="c")
coef(fit, matrix=TRUE)

# Both location and scale unknown
n = 2000
x = runif(n)
y = rlogis(n, loc=1+5*x, scale=exp(0+1*x))
fit = vglm(y ~ x, logistic2)
coef(fit, matrix=TRUE)
vcov(fit)
summary(fit)
```

---

logit *Logit Link Function*

Description

Computes the logit transformation, including its inverse and the first two derivatives.

Usage

```
logit(theta, earg = list(), inverse = FALSE, deriv = 0,
      short = TRUE, tag = FALSE)
elogit(theta, earg = list(min=0, max=1), inverse = FALSE, deriv = 0,
      short = TRUE, tag = FALSE)
```

Arguments

theta	Numeric or character. See below for further details.
earg	Optional list. Extra argument for passing in additional information. Values of theta which are less than or equal to 0 can be replaced by the bvalue component of the list earg before computing the link function value. Values of theta which are greater than or equal to 1 can be replaced by 1 minus the bvalue component of the list earg before computing the link function value. The component name bvalue stands for "boundary value". See Links for general information about earg. Similarly, for elogit, values of theta less than or equal to A or greater than or equal to B can be replaced by the bminvalue and bmaxvalue components of the list earg.

	For `elogit`, `earg` should be a list with components `min` giving A, `max` giving B, and for out of range values, `bminvalue` and `bmaxvalue`. If `earg` is used, these component names should not be abbreviated.
`inverse`	Logical. If `TRUE` the inverse function is computed. The inverse logit function is known as the *expit* function.
`deriv`	Order of the derivative. Integer with value 0, 1 or 2.
`short`	Used for labelling the `blurb` slot of a `vglmff-class` object.
`tag`	Used for labelling the linear/additive predictor in the `initialize` slot of a `vglmff-class` object. Contains a little more information if `TRUE`.

Details

The logit link function is very commonly used for parameters that lie in the unit interval. Numerical values of `theta` close to 0 or 1 or out of range result in `Inf`, `-Inf`, `NA` or `NaN`.

The *extended* logit link function `elogit` should be used more generally for parameters that lie in the interval (A, B), say. The formula is

$$\log((\theta - A)/(B - \theta))$$

and the default values for A and B correspond to the ordinary logit function. Numerical values of `theta` close to A or B or out of range result in `Inf`, `-Inf`, `NA` or `NaN`. However these can be replaced by values $bminvalue$ and $bmaxvalue$ first before computing the link function.

The arguments `short` and `tag` are used only if `theta` is character.

Value

For `logit` with `deriv = 0`, the logit of `theta`, i.e., `log(theta/(1-theta))` when `inverse = FALSE`, and if `inverse = TRUE` then `exp(theta)/(1+exp(theta))`.

For `deriv = 1`, then the function returns *d* `theta` / *d* `eta` as a function of `theta` if `inverse = FALSE`, else if `inverse = TRUE` then it returns the reciprocal.

Here, all logarithms are natural logarithms, i.e., to base e.

Note

Numerical instability may occur when `theta` is close to 1 or 0 (for `logit`), or close to A or B for `elogit`. One way of overcoming this is to use `earg`.

In terms of the threshold approach with cumulative probabilities for an ordinal response this link function corresponds to the univariate logistic distribution (see `logistic`).

Author(s)

Thomas W. Yee

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

See Also

Links, probit, cloglog, cauchit, loge.

Examples

```
p = seq(0.01, 0.99, by=0.01)
logit(p)
max(abs(logit(logit(p), inverse=TRUE) - p)) # Should be 0

p = c(seq(-0.02, 0.02, by=0.01), seq(0.97, 1.02, by=0.01))
logit(p)  # Has NAs
logit(p, earg=list(bvalue= .Machine$double.eps))  # Has no NAs

p = seq(0.9, 2.2, by=0.1)
elogit(p, earg=list(min=1, max=2,
                    bminvalue = 1 + .Machine$double.eps,
                    bmaxvalue = 2 - .Machine$double.eps))  # Has no NAs

## Not run:
par(mfrow=c(2,2))
y = seq(-4, 4, length=100)
for(d in 0:1) {
    matplot(p, cbind(logit(p, deriv=d), probit(p, deriv=d)),
            type="n", col="purple", ylab="transformation",
            lwd=2, las=1,
            main=if(d==0) "Some probability link functions"
            else "First derivative")
    lines(p, logit(p, deriv=d), col="limegreen", lwd=2)
    lines(p, probit(p, deriv=d), col="purple", lwd=2)
    lines(p, cloglog(p, deriv=d), col="chocolate", lwd=2)
    lines(p, cauchit(p, deriv=d), col="tan", lwd=2)
    if(d==0) {
        abline(v=0.5, h=0, lty="dashed")
        legend(0, 4.5, c("logit", "probit", "cloglog", "cauchit"),
               col=c("limegreen","purple","chocolate", "tan"), lwd=2)
    } else
        abline(v=0.5, lty="dashed")
}

for(d in 0) {
    matplot(y, cbind(logit(y, deriv=d, inverse=TRUE),
                     probit(y, deriv=d, inverse=TRUE)),
            type="n", col="purple", xlab="transformation", ylab="p",
            lwd=2, las=1,
            main=if(d==0) "Some inverse probability link functions"
            else "First derivative")
    lines(y, logit(y, deriv=d, inverse=TRUE), col="limegreen", lwd=2)
    lines(y, probit(y, deriv=d, inverse=TRUE), col="purple", lwd=2)
    lines(y, cloglog(y, deriv=d, inverse=TRUE), col="chocolate", lwd=2)
    lines(y, cauchit(y, deriv=d, inverse=TRUE), col="tan", lwd=2)
    if(d==0) {
        abline(h=0.5, v=0, lty="dashed")
```

```
        legend(-4, 1, c("logit", "probit", "cloglog", "cauchit"),
               col=c("limegreen","purple","chocolate", "tan"), lwd=2)
    }
}

p = seq(0.21, 0.59, by=0.01)
plot(p, elogit(p, earg=list(min=0.2, max=0.6)), lwd=2,
     type="l", col="black", ylab="transformation", xlim=c(0,1),
     las=1, main="elogit(p, earg=list(min=0.2, max=0.6)")
## End(Not run)
```

`loglinb2` *Loglinear Model for Two Binary Responses*

Description

Fits a loglinear model to two binary responses.

Usage

```
loglinb2(exchangeable = FALSE, zero = NULL)
```

Arguments

`exchangeable`	Logical. If `TRUE`, the two marginal probabilities are constrained to be equal. Should be set `TRUE` for ears, eyes, etc. data.
`zero`	Which linear/additive predictor is modelled as an intercept only? A `NULL` means none of them.

Details

The model is

$$P(Y_1 = y_1, Y_2 = y_2) = \exp(u_0 + u_1 y_1 + u_2 y_2 + u_{12} y_1 y_2)$$

where y_1 and y_2 are 0 or 1, and the parameters are u_1, u_2, u_{12}. The normalizing parameter u_0 can be expressed as a function of the other parameters, viz.,

$$u_0 = -\log[1 + \exp(u_1) + \exp(u_2) + \exp(u_1 + u_2 + u_{12})].$$

The linear/additive predictors are $(\eta_1, \eta_2, \eta_3)^T = (u_1, u_2, u_{12})^T$.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

When fitted, the `fitted.values` slot of the object contains the four joint probabilities, labelled as (Y_1, Y_2) = (0,0), (0,1), (1,0), (1,1), respectively.

Note

The response must be a two-column matrix of ones and zeros only. This is more restrictive than binom2.or, which can handle more types of input formats. Note that each of the 4 combinations of the multivariate response need to appear in the data set.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Wild, C. J. (2001) Discussion to: "Smoothing spline ANOVA for multivariate Bernoulli observations, with application to ophthalmology data (with discussion)" by Gao, F., Wahba, G., Klein, R., Klein, B. *Journal of the American Statistical Association*, **96**, 127–160.

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

binom2.or, binom2.rho, loglinb3.

Examples

```
data(coalminers)
coalminers = transform(coalminers, age=(age-42)/5)

# Get the n x 4 matrix of counts
temp = vglm(cbind(nBnW,nBW,BnW,BW) ~ age, binom2.or, coalminers)
counts = round(c(weights(temp, type="prior")) * temp@y)

# Create a n x 2 matrix response for loglinb2()
fred = matrix(c(0,0, 0,1, 1,0, 1,1), 4, 2, byrow=TRUE)
yy = kronecker(matrix(1, nrow(counts), 1), fred)
wt = c(t(counts))
age = rep(coalminers$age, rep(4, length(coalminers$age)))
yy = yy[wt>0,]
age = age[wt>0]
wt = wt[wt>0]

fit = vglm(yy ~ age, loglinb2, trace=TRUE, wei=wt)
coef(fit, mat=TRUE)    # Same! (at least for the log odds-ratio)
summary(fit)

# Try reconcile this with McCullagh and Nelder (1989), p.234
(0.166-0.131) / 0.027458   # 1.275 is approximately 1.25
```

`loglinb3` *Loglinear Model for Three Binary Responses*

Description

Fits a loglinear model to three binary responses.

Usage

```
loglinb3(exchangeable = FALSE, zero = NULL)
```

Arguments

`exchangeable`	Logical. If `TRUE`, the three marginal probabilities are constrained to be equal.
`zero`	Which linear/additive predictor is modelled as an intercept only? A `NULL` means none.

Details

The model is $P(Y_1 = y_1, Y_2 = y_2, Y_3 = y_3) =$

$$\exp(u_0 + u_1 y_1 + u_2 y_2 + u_3 y_3 + u_{12} y_1 y_2 + u_{13} y_1 y_3 + u_{23} y_2 y_3)$$

where y_1, y_2 and y_3 are 0 or 1, and the parameters are u_1, u_2, u_3, u_{12}, u_{13}, u_{23}. The normalizing parameter u_0 can be expressed as a function of the other parameters. Note that a third-order association parameter, u_{123} for the product $y_1 y_2 y_3$, is assumed to be zero for this family function.

The linear/additive predictors are $(\eta_1, \eta_2, \ldots, \eta_6)^T = (u_1, u_2, u_3, u_{12}, u_{13}, u_{23})^T$.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

When fitted, the `fitted.values` slot of the object contains the eight joint probabilities, labelled as (Y_1, Y_2, Y_3) = (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1), respectively.

Note

The response must be a three-column matrix of ones and zeros only. Note that each of the 8 combinations of the multivariate response need to appear in the data set, therefore data sets will need to be large in order for this family function to work.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Wild, C. J. (2001) Discussion to: "Smoothing spline ANOVA for multivariate Bernoulli observations, with application to ophthalmology data (with discussion)" by Gao, F., Wahba, G., Klein, R., Klein, B. *Journal of the American Statistical Association*, **96**, 127–160.

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

loglinb2, hunua.

Examples

```
data(hunua)
fit = vglm(cbind(cyadea,beitaw,kniexc) ~ altitude, loglinb3, data=hunua)
coef(fit, mat=TRUE)
fitted(fit)[1:4,]
summary(fit)
```

---

loglog *Log-log Link Function*

Description

Computes the log-log transformation, including its inverse and the first two derivatives.

Usage

```
loglog(theta, earg = list(), inverse = FALSE, deriv = 0,
       short = TRUE, tag = FALSE)
```

Arguments

theta	Numeric or character. See below for further details.
earg	Optional list. Extra argument for passing in additional information. Values of theta which are less than or equal to 1 can be replaced by the bvalue component of the list earg before computing the link function value. The component name bvalue stands for "boundary value". See Links for general information about earg.
inverse	Logical. If TRUE the inverse function is computed.
deriv	Order of the derivative. Integer with value 0, 1 or 2.
short	Used for labelling the blurb slot of a vglmff-class object.
tag	Used for labelling the linear/additive predictor in the initialize slot of a vglmff-class object. Contains a little more information if TRUE.

Details

The log-log link function is commonly used for parameters that are greater than unity. Numerical values of `theta` close to 1 or out of range result in `Inf`, `-Inf`, `NA` or `NaN`. The arguments `short` and `tag` are used only if `theta` is character.

Value

For `deriv = 0`, the log of `theta`, i.e., `log(log(theta))` when `inverse = FALSE`, and if `inverse = TRUE` then `exp(exp(theta))`.

For `deriv = 1`, then the function returns d `theta` / d `eta` as a function of `theta` if `inverse = FALSE`, else if `inverse = TRUE` then it returns the reciprocal.

Here, all logarithms are natural logarithms, i.e., to base e.

Note

Numerical instability may occur when `theta` is close to 1 unless `earg` is used.

Author(s)

Thomas W. Yee

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

See Also

`Links`, `loge`, `logoff`.

Examples

```
x = seq(0.8, 1.5, by=0.1)
loglog(x)  # Has NAs
loglog(x, earg=list(bvalue=1.0 + .Machine$double.eps))  # Has no NAs

x = seq(1.01, 10, len=100)
loglog(x)
max(abs(loglog(loglog(x), inverse=TRUE) - x)) # Should be 0
```

`lognormal` *Lognormal Distribution*

Description

Maximum likelihood estimation of the (univariate) lognormal distribution.

Usage

```
lognormal(lmeanlog = "identity", lsdlog = "loge",
          emeanlog=list(), esdlog=list(), zero = NULL)
lognormal3(lmeanlog = "identity", lsdlog = "loge",
           emeanlog=list(), esdlog=list(),
           powers.try = (-3):3, delta = NULL, zero = NULL)
```

Arguments

`lmeanlog, lsdlog`
: Parameter link functions applied to the mean and (positive) σ (standard deviation) parameter. Both of these are on the log scale. See `Links` for more choices.

`emeanlog, esdlog`
: List. Extra argument for each of the links. See `earg` in `Links` for general information.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. For `lognormal()`, the values must be from the set {1,2} which correspond to `mu`, `sigma`, respectively. For `lognormal3()`, the values must be from the set {1,2,3} where 3 is for λ.

`powers.try`
: Numerical vector. The initial *lambda* is chosen as the best value from `min(y) - 10^powers.try` where `y` is the response.

`delta`
: Numerical vector. An alternative method for obtaining an initial *lambda*. Here, `delta = min(y)-lambda`. If given, this supersedes the `powers.try` argument. The value must be positive.

Details

A random variable Y has a 2-parameter lognormal distribution if $\log(Y)$ is distributed $N(\mu, \sigma^2)$. The expected value of Y, which is

$$E(Y) = \exp(\mu + 0.5\sigma^2)$$

and not μ, make up the fitted values.

A random variable Y has a 3-parameter lognormal distribution if $\log(Y-\lambda)$ is distributed $N(\mu, \sigma^2)$. Here, $\lambda < Y$. The expected value of Y, which is

$$E(Y) = \lambda + \exp(\mu + 0.5\sigma^2)$$

and not μ, make up the fitted values.

`lognormal()` and `lognormal3()` fit the 2- and 3-parameter lognormal distribution respectively. Clearly, if the location parameter $\lambda = 0$ then both distributions coincide.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

See Also

rlnorm, normal1.

Examples

```
y = rlnorm(n <- 1000, meanlog=1.5, sdlog=exp(-0.8))
fit = vglm(y ~ 1, lognormal, trace=TRUE)
coef(fit, mat=TRUE)
Coef(fit)

x = runif(n <- 1000)
y = rlnorm(n, mean=0.5, sd=exp(x))
fit = vglm(y ~ x, lognormal(zero=1), trace=TRUE, crit="c")
coef(fit, mat=TRUE)
Coef(fit)

lambda = 4
y = lambda + rlnorm(n <- 1000, mean=1.5, sd=exp(-0.8))
fit = vglm(y ~ 1, lognormal3, trace=TRUE)
fit = vglm(y ~ 1, lognormal3, trace=TRUE, crit="c")
coef(fit, mat=TRUE)
Coef(fit)
summary(fit)
```

---

logoff *Log link function with an offset*

Description

Computes the log transformation with an offset, including its inverse and the first two derivatives.

Usage

```
logoff(theta, earg = list(offset=0), inverse = FALSE, deriv = 0,
       short = TRUE, tag = FALSE)
```

Arguments

theta	Numeric or character. See below for further details.
earg	List. Extra argument for passing in additional information. The offset component of the list earg is the offset value. See Links for general information about earg.
inverse	Logical. If TRUE the inverse function is computed.
deriv	Order of the derivative. Integer with value 0, 1 or 2.
short	Used for labelling the blurb slot of a vglmff-class object.
tag	Used for labelling the linear/additive predictor in the initialize slot of a vglmff-class object. Contains a little more information if TRUE.

Details

The log-offset link function is very commonly used for parameters that are greater than a certain value. In particular, it is defined by log(theta+offset) where offset is the offset value. For example, if offset=0.5 then the value of theta is restricted to be greater than -0.5.

Numerical values of theta close to -offset or out of range result in Inf, -Inf, NA or NaN. The arguments short and tag are used only if theta is character.

Value

For deriv = 0, the log of theta+offset, i.e., log(theta+offset) when inverse = FALSE, and if inverse = TRUE then exp(theta)-offset.

For deriv = 1, then the function returns d theta / d eta as a function of theta if inverse = FALSE, else if inverse = TRUE then it returns the reciprocal.

Here, all logarithms are natural logarithms, i.e., to base e.

Note

The default means this function is identical to loge.

Numerical instability may occur when theta is close to -offset.

Author(s)

Thomas W. Yee

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

See Also

Links, loge.

Examples

```
## Not run:
logoff(seq(-0.2, 0.5, by=0.1))
logoff(seq(-0.2, 0.5, by=0.1), earg=list(offset=0.5))
log(seq(-0.2, 0.5, by=0.1) + 0.5)
## End(Not run)
```

---

lomax *Lomax Distribution Family Function*

Description

Maximum likelihood estimation of the 2-parameter Lomax distribution.

Usage

```
lomax(link.scale = "loge", link.q = "loge",
      earg.scale=list(), earg.q=list(),
      init.scale = NULL, init.q = 1, zero = NULL)
```

Arguments

`link.scale, link.q`
: Parameter link function applied to the (positive) parameters `scale` and `q`. See `Links` for more choices.

`earg.scale, earg.q`
: List. Extra argument for each of the links. See `earg` in `Links` for general information.

`init.scale, init.q`
: Optional initial values for `scale` and `q`.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. Here, the values must be from the set {1,2} which correspond to `scale`, `q`, respectively.

Details

The 2-parameter Lomax distribution is the 4-parameter generalized beta II distribution with shape parameters $a = p = 1$. It is probably more widely known as the Pareto (II) distribution. It is also the 3-parameter Singh-Maddala distribution with shape parameter $a = 1$, as well as the beta distribution of the second kind with $p = 1$. More details can be found in Kleiber and Kotz (2003).

The Lomax distribution has density

$$f(y) = q/[b\{1 + y/b\}^{1+q}]$$

for $b > 0$, $q > 0$, $y > 0$. Here, b is the scale parameter `scale`, and `q` is a shape parameter. The cumulative distribution function is

$$F(y) = 1 - [1 + (y/b)]^{-q}.$$

The mean is

$$E(Y) = b/(q-1)$$

provided $q > 1$.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

If the self-starting initial values fail, try experimenting with the initial value arguments, especially those whose default value is not `NULL`.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

See Also

`Lomax`, `genbetaII`, `betaII`, `dagum`, `sinmad`, `fisk`, `invlomax`, `paralogistic`, `invparalogistic`.

Examples

```
y = rlomax(n=2000, 6, 2)
fit = vglm(y ~ 1, lomax, trace=TRUE)
fit = vglm(y ~ 1, lomax, trace=TRUE, crit="c")
coef(fit, mat=TRUE)
Coef(fit)
summary(fit)
```

---

lv *Latent Variables*

Description

Generic function for the *latent variables* of a model.

Usage

```
lv(object, ...)
```

Arguments

object	An object for which the extraction of latent variables is meaningful.
...	Other arguments fed into the specific methods function of the model. Sometimes they are fed into the methods function for Coef.

Details

Latent variables occur in reduced-rank regression models, as well as in quadratic and additive ordination models. For the latter two, latent variable values are often called *site scores* by ecologists. Latent variables are linear combinations of the explanatory variables.

Value

The value returned depends specifically on the methods function invoked.

Note

Latent variables are not really applicable to vglm/vgam models.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

Yee, T. W. (2004) A new technique for maximum-likelihood canonical Gaussian ordination. *Ecological Monographs*, **74**, 685–701.

Yee, T. W. (2006) Constrained additive ordination. *Ecology*, **87**, 203–213.

See Also

lv.qrrvglm, lv.rrvglm, lv.cao, lvplot.

Examples

```
## Not run:
data(hspider)
hspider[,1:6] = scale(hspider[,1:6]) # Standardized environmental vars
set.seed(123)
p1 = cao(cbind(Pardlugu, Pardmont, Pardnigr, Pardpull, Zoraspin) ~
         WaterCon + BareSand + FallTwig +
         CoveMoss + CoveHerb + ReflLux,
         family = poissonff, data = hspider,
         Rank = 1, df1.nl = c(Zoraspin=2.5, 3),
         Bestof = 3, Crow1positive = TRUE)

var(lv(p1))  # Scaled to unit variance  # Scaled to unit variance
c(lv(p1))    # Estimated site scores
```

```
## End(Not run)
```

`lvplot` *Latent Variable Plot*

Description

Generic function for a *latent variable plot* (also known as an *ordination diagram* by ecologists).

Usage

```
lvplot(object, ...)
```

Arguments

`object`	An object for a latent variable plot is meaningful.
`...`	Other arguments fed into the specific methods function of the model. They usually are graphical parameters, and sometimes they are fed into the methods function for `Coef`.

Details

Latent variables occur in reduced-rank regression models, as well as in quadratic and additive ordination. For the latter, latent variables are often called the *site scores*. Latent variable plots were coined by Yee (2004), and have the latent variable as at least one of its axes.

Value

The value returned depends specifically on the methods function invoked.

Note

Latent variables are not really applicable to `vglm`/`vgam` models.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2004) A new technique for maximum-likelihood canonical Gaussian ordination. *Ecological Monographs*, **74**, 685–701.

Yee, T. W. (2006) Constrained additive ordination. *Ecology*, **87**, 203–213.

See Also

`lvplot.qrrvglm`, `lvplot.cao`, `lv`, `trplot`.

Examples

```
## Not run:
data(hspider)
hspider[,1:6] = scale(hspider[,1:6]) # Standardized environmental vars
set.seed(123)
p1 = cao(cbind(Pardlugu, Pardmont, Pardnigr, Pardpull, Zoraspin) ~
         WaterCon + BareSand + FallTwig +
         CoveMoss + CoveHerb + ReflLux,
         family = poissonff, data = hspider, Bestof = 3,
         df1.nl = c(Zoraspin=2.5, 3), Crow1positive = TRUE)
index = 1:ncol(p1@y)
lvplot(p1, lcol=index, pcol=index, y=TRUE, las=1)

## End(Not run)
```

---

lvplot.qrrvglm *Latent Variable Plot for QO models*

Description

Produces an ordination diagram (latent variable plot) for quadratic ordination (QO) models. For rank-1 models, the x-axis is the first ordination/constrained/canonical axis. For rank-2 models, the x- and y-axis are the first and second ordination axes respectively.

Usage

```
lvplot.qrrvglm(object, varlvI = FALSE, reference = NULL,
    add = FALSE, plot.it = TRUE,
    rug = TRUE, y = FALSE, type = c("fitted.values", "predictors"),
    xlab = paste("Latent Variable", if (Rank == 1) "" else " 1", sep = ""),
    ylab = if (Rank == 1) switch(type, predictors = "Predictors",
    fitted.values = "Fitted values") else "Latent Variable 2",
    pcex = par()$cex, pcol = par()$col, pch = par()$pch,
    llty = par()$lty, lcol = par()$col, llwd = par()$lwd,
    label.arg = FALSE, adj.arg = -0.1,
    ellipse = 0.95, Absolute = FALSE,
    elty = par()$lty, ecol = par()$col, elwd = par()$lwd, egrid = 200,
    chull.arg = FALSE, clty = 2, ccol = par()$col, clwd = par()$lwd,
    cpch = "   ",
    C = FALSE, OriginC = c("origin", "mean"),
    Clty = par()$lty, Ccol = par()$col, Clwd = par()$lwd,
    Ccex = par()$cex, Cadj.arg = -0.1, stretchC = 1,
    sites = FALSE, spch = NULL, scol = par()$col, scex = par()$cex,
    sfont = par()$font, check.ok = TRUE, ...)
```

Arguments

object	A CQO or UQO object.
varlvI	Logical that is fed into Coef.qrrvglm.
reference	Integer or character that is fed into Coef.qrrvglm.
add	Logical. Add to an existing plot? If FALSE, a new plot is made.
plot.it	Logical. Plot it?
rug	Logical. If TRUE, a rug plot is plotted at the foot of the plot (applies to rank-1 models only). These values are jittered to expose ties.
y	Logical. If TRUE, the responses will be plotted (applies only to rank-1 models and if type="fitted.values".)
type	Either "fitted.values" or "predictors", specifies whether the y-axis is on the response or eta-scales respectively.
xlab	Caption for the x-axis. See par.
ylab	Caption for the y-axis. See par.
pcex	Character expansion of the points. Here, for rank-1 models, points are the response y data. For rank-2 models, points are the optima. See the cex argument in par.
pcol	Color of the points. See the col argument in par.
pch	Either an integer specifying a symbol or a single character to be used as the default in plotting points. See par. The pch argument can be of length M, the number of species.
llty	Line type. Rank-1 models only. See the lty argument of par.
lcol	Line color. Rank-1 models only. See the col argument of par.
llwd	Line width. Rank-1 models only. See the lwd argument of par.
label.arg	Logical. Label the optima and **C**? (applies only to rank-2 models only).
adj.arg	Justification of text strings for labelling the optima (applies only to rank-2 models only). See the adj argument of par.
ellipse	Numerical, of length 0 or 1 (applies only to rank-2 models only). If Absolute is TRUE then ellipse should be assigned a value that is used for the elliptical contouring. If Absolute is FALSE then ellipse should be assigned a value between 0 and 1, for example, setting ellipse=0.9 means an ellipse with contour = 90% of the maximum will be plotted about each optimum. If ellipse is a negative value, then the function checks that the model is an equal-tolerances model and varlvI=FALSE, and if so, plots circles with radius -ellipse. For example, setting ellipse=-1 will result in circular contours that have unit radius (in latent variable units). If ellipse is NULL or FALSE then no ellipse is drawn around the optima.
Absolute	Logical. If TRUE, the contours corresponding to ellipse are on an absolute scale. If FALSE, the contours corresponding to ellipse are on a relative scale.
elty	Line type of the ellipses. See the lty argument of par.
ecol	Line color of the ellipses. See the col argument of par.

elwd	Line width of the ellipses. See the lwd argument of par.
egrid	Numerical. Line resolution of the ellipses. Choosing a larger value will result in smoother ellipses. Useful when ellipses are large.
chull.arg	Logical. Add a convex hull around the site scores?
clty	Line type of the convex hull. See the lty argument of par.
ccol	Line color of the convex hull. See the col argument of par.
clwd	Line width of the convex hull. See the lwd argument of par.
cpch	Character to be plotted at the intersection points of the convex hull. Having white spaces means that site labels are not obscured there. See the pch argument of par.
C	Logical. Add **C** (represented by arrows emanating from OriginC) to the plot?
OriginC	Character or numeric. Where the arrows representing **C** emanate from. If character, it must be one of the choices given. By default the first is chosen. The value "origin" means c(0,0). The value "mean" means the sample mean of the latent variables (centroid). Alternatively, the user may specify a numerical vector of length 2.
Clty	Line type of the arrows representing **C**. See the lty argument of par.
Ccol	Line color of the arrows representing **C**. See the col argument of par.
Clwd	Line width of the arrows representing **C**. See the lwd argument of par.
Ccex	Numeric. Character expansion of the labelling of **C**. See the cex argument of par.
Cadj.arg	Justification of text strings when labelling **C**. See the adj argument of par.
stretchC	Numerical. Stretching factor for **C**. Instead of using **C**, stretchC * **C** is used.
sites	Logical. Add the site scores (aka latent variable values, nu's) to the plot? (applies only to rank-2 models only).
spch	Plotting character of the site scores. The default value of NULL means the row labels of the data frame are used. They often are the site numbers. See the pch argument of par.
scol	Color of the site scores. See the col argument of par.
scex	Character expansion of the site scores. See the cex argument of par.
sfont	Font used for the site scores. See the font argument of par.
check.ok	Logical. Whether a check is performed to see that Norrr = ~ 1 was used. It doesn't make sense to have a latent variable plot unless this is so.
...	Arguments passed into the plot function when setting up the entire plot. Useful arguments here include xlim and ylim.

Details

This function only works for rank-1 and rank-2 QRR-VGLMs with argument Norrr = ~ 1.

For unequal-tolerances models, the latent variable axes can be rotated so that at least one of the tolerance matrices is diagonal; see Coef.qrrvglm for details.

Arguments beginning with "p" correspond to the points e.g., pcex and pcol correspond to the size and color of the points. Such "p" arguments should be vectors of length 1, or n, the number of sites. For the rank-2 model, arguments beginning with "p" correspond to the optima.

Value

Returns a matrix of latent variables (site scores) regardless of whether a plot was produced or not.

Warning

Interpretation of a latent variable plot (CQO diagram) is potentially very misleading in terms of distances if (i) the tolerance matrices of the species are unequal and (ii) the contours of these tolerance matrices are not included in the ordination diagram.

Note

A species which does not have an optimum will not have an ellipse drawn even if requested, i.e., if its tolerance matrix is not positive-definite.

Plotting **C** gives a visual display of the weights (loadings) of each of the variables used in the linear combination defining each latent variable.

The arguments elty, ecol and elwd, may be replaced in the future by llty, lcol and llwd, respectively.

For rank-1 models, a similar function to this one is persp.qrrvglm. It plots the fitted values on a more fine grid rather than at the actual site scores here. The result is a collection of smooth bell-shaped curves. However, it has the weakness that the plot is more divorced from the data; the user thinks it is the truth without an appreciation of the statistical variability in the estimates.

In the example below, the data comes from an equal-tolerances model. The species' tolerance matrices are all the identity matrix, and the optimums are at (0,0), (1,1) and (-2,0) for species 1, 2, 3 respectively.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2004) A new technique for maximum-likelihood canonical Gaussian ordination. *Ecological Monographs*, **74**, 685–701.

See Also

lvplot, persp.qrrvglm, Coef.qrrvglm, par, cqo.

Examples

```
set.seed(123)
x2 = rnorm(n <- 200)     # Has mean 0 (needed when ITol=TRUE)
x3 = rnorm(n)            # Has mean 0 (needed when ITol=TRUE)
x4 = rnorm(n)            # Has mean 0 (needed when ITol=TRUE)
```

```
lv1 =  x2 + x3 - 2*x4
lv2 = -x2 + x3 + 0*x4   # lv2 is weakly correlated with lv1
lambda1 = exp(6 - 0.5 * (lv1-0)^2 - 0.5 * (lv2-0)^2)
lambda2 = exp(5 - 0.5 * (lv1-1)^2 - 0.5 * (lv2-1)^2)
lambda3 = exp(5 - 0.5 * (lv1+2)^2 - 0.5 * (lv2-0)^2)
spp1 = rpois(n, lambda1)
spp2 = rpois(n, lambda2)
spp3 = rpois(n, lambda3)
set.seed(111)
p2 = cqo(cbind(spp1,spp2,spp3) ~ x2 + x3 + x4, poissonff,
         Rank=2, ITolerances=TRUE,
         Crow1positive=c(TRUE,FALSE))   # deviance = 505.81
if(deviance(p2) > 506) stop("suboptimal fit obtained")
sort(p2@misc$deviance.Bestof)  # A history of the fits
Coef(p2)

## Not run:
lvplot(p2, sites=TRUE, spch="*", scol="darkgreen", scex=1.5,
       chull=TRUE, label=TRUE, Absolute=TRUE, ellipse=140,
       adj=-0.5, pcol="blue", pcex=1.3, las=1,
       C=TRUE, Cadj=c(-.3,-.3,1), Clwd=2, Ccex=1.4, Ccol="red",
       main=paste("Contours at Abundance=140 with",
                  "convex hull of the site scores"))
## End(Not run)
var(lv(p2)) # A diagonal matrix, i.e., uncorrelated latent variables
var(lv(p2, varlvI=TRUE)) # Identity matrix
Tol(p2)[,,1:2] # Identity matrix
Tol(p2, varlvI=TRUE)[,,1:2] # A diagonal matrix
```

`lvplot.rrvglm` *Latent Variable Plot for RR-VGLMs*

Description

Produces an *ordination diagram* (also known as a *biplot* or *latent variable plot*) for *reduced-rank vector generalized linear models* (RR-VGLMs). For rank-2 models only, the x- and y-axis are the first and second canonical axes respectively.

Usage

```
lvplot.rrvglm(object,
              A = TRUE, C = TRUE, scores = FALSE, plot.it = TRUE,
              groups = rep(1, n), gapC = sqrt(sum(par()$cxy^2)),
              scaleA = 1,
              xlab = "Latent Variable 1", ylab = "Latent Variable 2",
              Alabels = if (length(object@misc$predictors.names))
              object@misc$predictors.names else paste("LP", 1:M, sep = ""),
              Aadj = par()$adj, Acex = par()$cex, Acol = par()$col,
              Apch = NULL,
```

```
Clabels = dimnames(Cmat)[[1]], Cadj = par()$adj,
Ccex = par()$cex, Ccol = par()$col, Clty = par()$lty,
Clwd = par()$lwd,
chull.arg = FALSE, ccex = par()$cex, ccol = par()$col,
clty = par()$lty, clwd = par()$lwd,
spch = NULL, scex = par()$cex, scol = par()$col,
slabels = dimnames(x2mat)[[1]], ...)
```

Arguments

object	Object of class "rrvglm".
A	Logical. Allow the plotting of **A**?
C	Logical. Allow the plotting of **C**? If TRUE then **C** is represented by arrows emenating from the origin.
scores	Logical. Allow the plotting of the n scores? The scores are the values of the latent variables for each observation.
plot.it	Logical. Plot it? If FALSE, no plot is produced and the matrix of scores (n latent variable values) is returned. If TRUE, the rank of object need not be 2.
groups	A vector whose distinct values indicate which group the observation belongs to. By default, all the observations belong to a single group. Useful for the multinomial logit model (see multinomial.
gapC	The gap between the end of the arrow and the text labelling of **C**, in latent variable units.
scaleA	Numerical value that is multiplied by **A**, so that **C** is divided by this value.
xlab	Caption for the x-axis. See par.
ylab	Caption for the y-axis. See par.
Alabels	Character vector to label **A**. Must be of length M.
Aadj	Justification of text strings for labelling **A**. See the adj argument of par.
Acex	Numeric. Character expansion of the labelling of **A**. See the cex argument of par.
Acol	Line color of the arrows representing **C**. See the col argument of par.
Apch	Either an integer specifying a symbol or a single character to be used as the default in plotting points. See par. The pch argument can be of length M, the number of species.
Clabels	Character vector to label **C**. Must be of length $p2$.
Cadj	Justification of text strings for labelling **C**. See the adj argument of par.
Ccex	Numeric. Character expansion of the labelling of **C**. See the cex argument of par.
Ccol	Line color of the arrows representing **C**. See the col argument of par.
Clty	Line type of the arrows representing **C**. See the lty argument of par.
Clwd	Line width of the arrows representing **C**. See the lwd argument of par.
chull.arg	Logical. Plot the convex hull of the scores? This is done for each group (see the group argument).

ccex	Numeric. Character expansion of the labelling of the convex hull. See the cex argument of par.
ccol	Line color of the convex hull. See the col argument of par.
clty	Line type of the convex hull. See the lty argument of par.
clwd	Line width of the convex hull. See the lwd argument of par.
spch	Either an integer specifying a symbol or a single character to be used as the default in plotting points. See par. The spch argument can be of length M, the number of species.
scex	Numeric. Character expansion of the labelling of the scores. See the cex argument of par.
scol	Line color of the arrows representing **C**. See the col argument of par.
slabels	Character vector to label the scores. Must be of length n.
...	Arguments passed into the plot function when setting up the entire plot. Useful arguments here include xlim and ylim.

Details

For RR-VGLMs, a *biplot* and a *latent variable* plot coincide. In general, many of the arguments starting with "A" refer to **A** (of length M), "C" to **C** (of length $p2$), "c" to the convex hull (of length length(unique(groups))), and "s" to scores (of length n).

As the result is a biplot, its interpretation is based on the inner product.

Value

The matrix of scores (n latent variable values) is returned regardless of whether a plot was produced or not.

Note

The functions lvplot.rrvglm and biplot.rrvglm are equivalent.

In the example below the predictor variables are centered, which is a good idea.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

See Also

lvplot, par, rrvglm, Coef.rrvglm, rrvglm.control.

Examples

```
data(pneumo)
n = nrow(pneumo)  # x1, x2 and x3 are some unrelated covariates
pneumo = transform(pneumo, slet=scale(log(exposure.time)),
                   x1 = rnorm(n), x2 = rnorm(n), x3 = rnorm(n))
fit = rrvglm(cbind(normal, mild, severe) ~ slet + x1 + x2 + x3,
             multinomial, pneumo, Rank=2, Corner=FALSE, Uncor=TRUE)
## Not run:
lvplot(fit, chull=TRUE, scores=TRUE, clty=2, ccol="blue", scol="red",
       Ccol="darkgreen", Clwd=2, Ccex=2,
       main="Biplot of some fictitional data")
## End(Not run)
```

---

maxwell *Maxwell Distribution Family Function*

Description

Estimating the parameter of the Maxwell distribution by maximum likelihood estimation.

Usage

```
maxwell(link = "loge", earg=list())
```

Arguments

link	Parameter link function applied to the parameter a. See Links for more choices. A log link is the default because the parameter is positive.
earg	List. Extra argument for the link. See earg in Links for general information.

Details

The Maxwell distribution, which is used in the area of thermodynamics, has a probability density function that can be written

$$f(y) = \sqrt{2/\pi} a^{3/2} y^2 \exp(-0.5ay^2)$$

for $y > 0$ and $a > 0$. The mean of Y is $\sqrt{8/(a\pi)}$ (returned as the fitted values), and its variance is $(3\pi - 8)/(\pi a)$.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Note

A related distribution is the Rayleigh distribution. Fisher-scoring and Newton-Raphson are the same here.

Author(s)

T. W. Yee

References

von Seggern, D. H. (1993) *CRC Standard Curves and Surfaces*, Boca Raton, FL.: CRC Press.

See Also

Maxwell, rayleigh.

Examples

```
y = rmaxwell(n <- 1000, a=exp(2))
fit = vglm(y ~ 1, maxwell, trace=TRUE, crit="c")
coef(fit, matrix=TRUE)
Coef(fit)
```

---

mccullagh89 *McCullagh (1989) Distribution Family Function*

Description

Estimates the two parameters of the McCullagh (1989) distribution by maximum likelihood estimation.

Usage

```
mccullagh89(ltheta="rhobit", lnu="logoff",
            itheta=NULL, inu=NULL,
            etheta=list(),
            enu=if(lnu == "logoff") list(offset=0.5) else list(),
            zero=NULL)
```

Arguments

ltheta, lnu	Link functions for the θ and ν parameters. See Links for more choices.
itheta, inu	Numeric. Optional initial values for θ and ν. The default is to internally compute them.
etheta, enu	List. Extra argument associated with ltheta and lnu containing any extra information. See Links for general information about **VGAM** link functions.
zero	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The default is none of them. If used, choose one value from the set {1,2}.

Details

The McCullagh (1989) distribution has density function

$$f(y;\theta,\nu) = \frac{\{1-y^2\}^{\nu-\frac{1}{2}}}{(1-2\theta y+\theta^2)^{\nu}\text{Beta}(\nu+\frac{1}{2},\frac{1}{2})}$$

where $-1 < y < 1$ and $-1 < \theta < 1$. This distribution is equation (1) in that paper. The parameter ν satisfies $\nu > -1/2$, therefore the default is to use an log-offset link with offset equal to 0.5, i.e., $\eta_2 = \log(\nu + 0.5)$. The mean is of Y is $\nu\theta/(1+\nu)$, and these are returned as the fitted values.

This distribution is related to the Leipnik distribution (see Johnson et al. (1995)), is related to ultraspherical functions, and under certain conditions, arises as exit distributions for Brownian motion. Fisher scoring is implemented here and it uses a diagonal matrix so the parameters are globally orthogonal in the Fisher information sense. McCullagh (1989) also states that, to some extent, θ and ν have the properties of a location parameter and a precision parameter, respectively.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

Note

Convergence may be slow or fail unless the initial values are reasonably close. If a failure occurs, try assigning the argument `inu` and/or `itheta`. Figure 1 of McCullagh (1989) gives a broad range of densities for different values of θ and ν, and this could be consulted for obtaining reasonable initial values if all else fails.

Author(s)

T. W. Yee

References

McCullagh, P. (1989) Some statistical properties of a family of continuous univariate distributions. *Journal of the American Statistical Association*, **84**, 125–129.

Johnson, N. L. and Kotz, S. and Balakrishnan, N. (1995) *Continuous Univariate Distributions*, 2nd edition, Volume 2, New York: Wiley. (pages 612–617).

See Also

`leipnik`, `rhobit`, `logoff`.

Examples

```
n = 1000
y = rnorm(n, mean=0.0, sd=0.2)  # Limit as theta is 0, nu is infinity
fit = vglm(y ~ 1, mccullagh89, trace=TRUE)
fitted(fit)[1:5]
mean(y)
```

```
summary(fit)
coef(fit, matrix=TRUE)
Coef(fit)
```

`mckaygamma2` *McKay's Bivariate Gamma Distribution*

Description

Estimate the two parameters of McKay's bivariate gamma distribution using maximum likelihood estimation.

Usage

```
mckaygamma2(la = "loge", lp = "loge", lq = "loge",
            ia = NULL, ip = 1, iq = 1, zero = NULL)
```

Arguments

`la, lp, lq`	Link functions applied to the (positive) parameters a, p and q. See `Links` for more choices.
`ia, ip, iq`	Initial values for a, p and q. The default for a is to estimate it using `ip` and `iq`.
`zero`	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,3}. The default is none of them.

Details

The joint probability density function is given by

$$f(y_1, y_2) = a^{p+q} y_1^{p-1} (y_2 - y_1)^{q-1} \exp(-a y_2)/[\Gamma(p)\Gamma(q)]$$

for $a > 0$, $p > 0$, $q > 0$ and $0 < y_1 < y_2$. Here, Γ is the gamma function, as in `gamma`. By default, the linear/additive predictors are $\eta_1 = \log(a)$, $\eta_2 = \log(p)$, $\eta_3 = \log(q)$.

Although Fisher scoring and Newton-Raphson coincide for this distribution, faster convergence may be obtained by choosing better values for the arguments `ip` and `iq`.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Note

The response must be a two column matrix. Currently, the fitted value is a matrix with two columns; the first column has values $(p + q)/a$ for the mean of `pmin(y1,y2)`, while the second column is filled with `NA` for the unknown mean of `pmax(y1,y2)`. The data are sorted internally and the user need not input the data presorted.

Author(s)

T. W. Yee

References

McKay, A. T. (1934) Sampling from batches. *Journal of the Royal Statistical Society—Supplement*, **1**, 207–216.

Kotz, S. and Balakrishnan, N. and Johnson, N. L. (2000) *Continuous Multivariate Distributions Volume 1: Models and Applications*, 2nd edition, New York: Wiley.

See Also

gamma2.

Examples

```
y1 = rgamma(n <- 200, shape=4)
y2 = rgamma(n, shape=8)
ymat = cbind(y1,y2)
fit = vglm(ymat ~ 1, fam=mckaygamma2, trace=TRUE)
coef(fit, matrix=TRUE)
Coef(fit)
vcov(fit)
fitted(fit)[1:5,]
summary(fit)
```

---

meplot *Mean Excess Plot*

Description

Mean excess plot (also known as a mean residual life plot), a diagnostic plot for the generalized Pareto distribution (GPD).

Usage

```
meplot(object, ...)
meplot.default(y, main="Mean Excess Plot",
    xlab="Threshold", ylab="Mean Excess", lty=c(2,1:2),
    conf=0.95, col=c("blue","black","blue"), type="l", ...)
meplot.vlm(object, ...)
```

Arguments

y	A numerical vector. NAs etc. are not allowed.
main	Character. Overall title for the plot.
xlab	Character. Title for the x axis.
ylab	Character. Title for the y axis.
lty	Line type. The second value is for the mean excess value, the first and third values are for the envelope surrounding the confidence interval.
conf	Confidence level. The default results in approximate 95 percent confidence intervals for each mean excess value.
col	Colour of the three lines.
type	Type of plot. The default means lines are joined between the mean excesses and also the upper and lower limits of the confidence intervals.
object	An object that inherits class "vlm", usually of class vglm-class or vgam-class.
...	Graphical argument passed into plot. See par for an exhaustive list. The arguments xlim and ylim are particularly useful.

Details

If Y has a GPD with scale parameter σ and shape parameter $\xi < 1$, and if $y > 0$, then

$$E(Y - u|Y > u) = \frac{\sigma + \xi u}{1 - \xi}.$$

It is a linear function in u, the threshold. Note that $Y - u$ is called the *excess* and values of Y greater than u are called *exceedences*. The empirical versions used by these functions is to use sample means to estimate the left hand side of the equation. Values of u in the plot are the values of y itself. If the plot is roughly a straight line then the GPD is a good fit; this plot can be used to select an appropriate threshold value. See gpd for more details. If the plot is flat then the data may be exponential, and if it is curved then it may be Weibull or gamma.

The function meplot is generic, and meplot.default and meplot.vlm are some methods functions for mean excess plots.

Value

A list is returned invisibly with the following components.

threshold	The x axis values.
meanExcess	The y axis values. Each value is a sample mean minus a value u.

Note

The function is designed for speed and not accuracy, therefore huge data sets with extremely large values may cause failure (the function cumsum is used.) Ties may not be well handled.

Author(s)

T. W. Yee

References

Davison, A. C. and Smith, R. L. (1990) Models for exceedances over high thresholds (with discussion). *Journal of the Royal Statistical Society, Series B, Methodological*, **52**, 393–442.

Coles, S. (2001) *An Introduction to Statistical Modeling of Extreme Values*. London: Springer-Verlag.

See Also

gpd.

Examples

```
## Not run:
meplot(runif(500), las=1) -> i
names(i)
## End(Not run)
```

---

micmen *Michaelis-Menten Model*

Description

Fits a Michaelis-Menten nonlinear regression model.

Usage

```
micmen(rpar = 0.001, divisor = 10, init1 = NULL, init2 = NULL,
       link1 = "identity", link2 = "identity",
       earg1=list(), earg2=list(),
       dispersion = 0, zero = NULL)
```

Arguments

rpar	Numeric. Initial positive ridge parameter. This is used to create positive-definite weight matrices.
divisor	Numerical. The divisor used to divide the ridge parameter at each iteration until it is very small but still positive. The value of divisor should be greater than one.
init1, init2	Numerical. Initial value for the first and second parameters, respectively. The default is to use a self-starting value.
link1, link2	Parameter link function applied to the first and second parameters, respectively. See Links for more choices.

`earg1, earg2` List. Extra argument for each of the links. See `earg` in `Links` for general information.

`dispersion` Numerical. Dispersion parameter.

`zero` An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2}. A `NULL` means none.

Details

The Michaelis-Menten Model is given by

$$E(Y_i) = (\theta_1 x_i)/(\theta_2 + x_i)$$

where θ_1 and θ_2 are the two parameters.

The relationship between iteratively reweighted least squares and the Gauss-Newton algorithm is given in Wedderburn (1974). However, the algorithm used by this family function is different. Details are given at the Author's web site.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

The regressor values x_i are inputted as a vector in the `regressor` argument. To predict the response at new values of x_i one must assign the `@extra$uvec` slot in the fitted object these values, e.g., see the example below.

Numerical problems may occur. If so, try setting some initial values for the parameters. In the future, several self-starting initial values will be implemented.

Author(s)

T. W. Yee

References

Seber, G. A. F. and Wild, C. J. (1989) *Nonlinear Regression*, New York: Wiley.

Wedderburn, R. W. M. (1974) Quasi-likelihood functions, generalized linear models, and the Gauss-Newton method. *Biometrika*, **61**, 439–447.

Bates, D. M. and Watts, D. G. (1988) *Nonlinear Regression Analysis and Its Applications*, New York: Wiley.

Documentation accompanying the **VGAM** package at `http://www.stat.auckland.ac.nz/~yee` contains further information and examples.

See Also

`enzyme`.

Examples

```
data(enzyme)
fit = vglm(velocity ~ 1, micmen, enzyme, trace=TRUE, crit="c",
           regressor=enzyme$conc)
## Not run:
attach(enzyme)
plot(conc, velocity, xlab="concentration", las=1, main="Enzyme data")
lines(conc, fitted(fit), col="blue") # Join up the fitted values
detach(enzyme)

# Predict the response at other concentrations and add it to the plot
newdata = new=data.frame(concentration=seq(0, 2, len=200))
fit@extra$uvec = newdata$concentration
lines(newdata$conc, predict(fit, new=newdata, type="res"), col="red")
## End(Not run)
summary(fit)
```

`mix2normal1` *Mixture of Two Univariate Normal Distributions*

Description

Estimates the five parameters of a mixture of two univariate normal distributions by maximum likelihood estimation.

Usage

```
mix2normal1(lphi="logit", lmu="identity", lsd="loge",
            ephi=list(), emu1=list(), emu2=list(), esd1=list(), esd2=list(),
            iphi=0.5, imu1=NULL, imu2=NULL, isd1=NULL, isd2=NULL,
            qmu=c(0.2, 0.8), esd=FALSE, zero=1)
```

Arguments

`lphi`	Link function for the parameter ϕ. See below for more details. See `Links` for more choices.
`lmu`	Link function applied to each μ parameter. See `Links` for more choices.
`lsd`	Link function applied to each σ parameter. See `Links` for more choices.
`ephi, emu1, emu2, esd1, esd2`	List. Extra argument for each of the links. See `earg` in `Links` for general information. If `esd=TRUE` then `esd1` is used and not `esd2`.
`iphi`	Initial value for ϕ, whose value must lie between 0 and 1.
`imu1, imu2`	Optional initial value for μ_1 and μ_2. The default is to compute initial values internally using the argument `qmu`.
`isd1, isd2`	Optional initial value for σ_1 and σ_2. The default is to compute initial values internally based on the argument `qmu`.

qmu — Vector with two values giving the probabilities relating to the sample quantiles for obtaining initial values for μ_1 and μ_2. The two values are fed in as the probs argument into quantile.

esd — Logical indicating whether the two standard deviations should be constrained to be equal. If set TRUE, the appropriate constraint matrices will be used.

zero — An integer specifying which linear/additive predictor is modelled as intercepts only. If given, the value or values must be from the set $\{1, 2, \ldots, 5\}$. The default is the first one only, meaning ϕ is a single parameter even when there are explanatory variables. Set zero=NULL to model all linear/additive predictors as functions of the explanatory variables.

Details

The probability function can be loosely written as

$$f(y) = \phi\, N(\mu_1, \sigma_1^2) + (1 - \phi)\, N(\mu_2, \sigma_2^2)$$

where ϕ is the probability an observation belongs to the first group. The parameters μ_1 and μ_2 are the means, and σ_1 and σ_2 are the standard deviations. The parameter ϕ satisfies $0 < \phi < 1$. The mean of Y is $\phi\mu_1 + (1 - \phi)\mu_2$ and this is returned as the fitted values. By default, the five linear/additive predictors are $(logit(\phi), \mu_1, \log(\sigma_1), \mu_2, \log(\sigma_2))^T$. If esd=TRUE then $\sigma_1 = \sigma_2$ is enforced.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

Numerical problems can occur. Half-stepping is not uncommon. If failure to converge occurs, try obtaining better initial values, e.g., by using iphi and qmu etc.

This function uses a quasi-Newton update for the working weight matrices (BFGS variant). It builds up approximations to the weight matrices, and currently the code is not fully tested. In particular, results based on the weight matrices (e.g., from vcov and summary) may be quite incorrect, especially when the arguments weights is used to input prior weights.

This **VGAM** family function should be used with caution.

Note

Fitting this model successfully to data can be difficult due to numerical problems and ill-conditioned data. It pays to fit the model several times with different initial values, and check that the best fit looks reasonable. Plotting the results is recommended. This function works better as μ_1 and μ_2 become more different.

Convergence is often slow, especially when the two component distributions are not well separated. The control argument maxit should be set to a higher value, e.g., 200, and use trace=TRUE to monitor convergence. If appropriate in the first place, setting esd=TRUE often makes the optimization problem much easier in general.

Author(s)

T. W. Yee

References

Everitt, B. S. and Hand, D. J. (1981) *Finite Mixture Distributions*. London: Chapman & Hall.

See Also

normal1, Normal, mix2poisson.

Examples

```
n = 1000
mu1 =  99   # Mean IQ of geography professors
mu2 = 150   # Mean IQ of mathematics professors
sd1 = sd2 = 16
phi = 0.3
y = ifelse(runif(n) < phi, rnorm(n, mu1, sd1), rnorm(n, mu2, sd2))

# Good idea to have trace=TRUE:
fit = vglm(y ~ 1, mix2normal1(esd=TRUE), maxit=200)
coef(fit, matrix=TRUE) # the estimates
c(phi, mu1, sd1, mu2, sd2) # the truth

## Not run:
# Plot the results
xx = seq(min(y), max(y), len=200)
plot(xx, (1-phi)*dnorm(xx, mu2, sd2), type="l", xlab="IQ",
     main="Red=estimate, blue=truth", col="blue")
phi.est = logit(coef(fit)[1], inverse=TRUE)
sd.est = exp(coef(fit)[3])
lines(xx, phi*dnorm(xx, mu1, sd1), col="blue")
lines(xx, phi.est * dnorm(xx, Coef(fit)[2], sd.est), col="red")
lines(xx, (1-phi.est) * dnorm(xx, Coef(fit)[4], sd.est), col="red")
abline(v=Coef(fit)[c(2,4)], lty=2, col="red")
abline(v=c(mu1, mu2), lty=2, col="blue")
## End(Not run)
```

---

mix2poisson *Mixture of Two Poisson Distributions*

Description

Estimates the three parameters of a mixture of two Poisson distributions by maximum likelihood estimation.

Usage

```
mix2poisson(lphi = "logit", llambda = "loge",
            ephi=list(), el1=list(), el2=list(),
            iphi = 0.5, il1 = NULL, il2 = NULL,
            qmu = c(0.2, 0.8), zero = 1)
```

Arguments

lphi	Link function for the parameter ϕ. See below for more details. See Links for more choices.
llambda	Link function applied to each λ parameter. See Links for more choices.
ephi, el1, el2	List. Extra argument for each of the links. See earg in Links for general information.
iphi	Initial value for ϕ, whose value must lie between 0 and 1.
il1, il2	Optional initial value for λ_1 and λ_2. These values must be positive. The default is to compute initial values internally using the argument qmu.
qmu	Vector with two values giving the probabilities relating to the sample quantiles for obtaining initial values for λ_1 and λ_2. The two values are fed in as the probs argument into quantile.
zero	An integer specifying which linear/additive predictor is modelled as intercepts only. If given, the value must be either 1 and/or 2 and/or 3, and the default is the first one only, meaning ϕ is a single parameter even when there are explanatory variables. Set zero=NULL to model all linear/additive predictors as functions of the explanatory variables.

Details

The probability function can be loosely written as

$$P(Y = y) = \phi \, Poisson(\lambda_1) + (1 - \phi) \, Poisson(\lambda_2)$$

where ϕ is the probability an observation belongs to the first group, and $y = 0, 1, 2, \ldots$. The parameter ϕ satisfies $0 < \phi < 1$. The mean of Y is $\phi\lambda_1 + (1 - \phi)\lambda_2$ and this is returned as the fitted values. By default, the three linear/additive predictors are $(logit(\phi), \log(\lambda_1), \log(\lambda_2))^T$.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

Numerical problems can occur. Half-stepping is not uncommon. If failure to converge occurs, try obtaining better initial values, e.g., by using iphi and qmu etc.

This function uses a quasi-Newton update for the working weight matrices (BFGS variant). It builds up approximations to the weight matrices, and currently the code is not fully tested. In particular,

results based on the weight matrices (e.g., from `vcov` and `summary`) may be quite incorrect, especially when the arguments `weights` is used to input prior weights.

This **VGAM** family function should be used with caution.

Note

Fitting this model successfully to data can be difficult due to numerical problems and ill-conditioned data. It pays to fit the model several times with different initial values, and check that the best fit looks reasonable. Plotting the results is recommended. This function works better as λ_1 and λ_2 become more different.

Convergence is often slow, especially when the two component distributions are not well separated. The control argument `maxit` should be set to a higher value, e.g., 200, and use `trace=TRUE` to monitor convergence.

Author(s)

T. W. Yee

See Also

`rpois`, `mix2normal1`.

Examples

```
n = 3000
mu1 = exp(2.4) # also known as lambda1
mu2 = exp(3.1)
phi = 0.3
y = ifelse(runif(n) < phi, rpois(n, mu1), rpois(n, mu2))

fit = vglm(y ~ 1, mix2poisson, maxit=200) # good idea to have trace=TRUE
coef(fit, matrix=TRUE)
Coef(fit) # the estimates
c(phi, mu1, mu2) # the truth

## Not run:
# Plot the results
ty = table(y)
plot(names(ty), ty, type="h", main="Red=estimate, blue=truth")
abline(v=Coef(fit)[-1], lty=2, col="red")
abline(v=c(mu1, mu2), lty=2, col="blue")
## End(Not run)
```

model.framevlm *Construct the Model Frame of a VLM Object*

Description

This function returns a data.frame with the variables. It is applied to an object which inherits from class "vlm" (e.g., a fitted model of class "vglm").

Usage

```
model.framevlm(object, ...)
```

Arguments

object	a model object from the **VGAM** R package that inherits from a *vector linear model* (VLM), e.g., a model of class "vglm".
...	further arguments such as data, na.action, subset. See model.frame for more information on these.

Details

Since object is an object which inherits from class "vlm" (e.g., a fitted model of class "vglm"), the method will either returned the saved model frame used when fitting the model (if any, selected by argument model = TRUE) or pass the call used when fitting on to the default method.

This code implements *smart prediction* (see smartpred).

Value

A data.frame containing the variables used in the object plus those specified in ....

References

Chambers, J. M. (1992) *Data for models.* Chapter 3 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

See Also

model.frame, model.matrixvlm, predict.vglm, smartpred.

Examples

```
# Illustrates smart prediction
data(pneumo)
pneumo = transform(pneumo, let=log(exposure.time))
fit = vglm(cbind(normal,mild, severe) ~ poly(c(scale(let)), 2),
           fam=multinomial,
           data=pneumo, trace=TRUE, x=FALSE)
class(fit)
```

```
check1 = model.frame(fit)[1:3,]
check1
check2 = model.frame(fit, data=pneumo[1:3,])
check2
all.equal(unlist(check1), unlist(check2)) # Should be TRUE

q0 = predict(fit)[1:3,]
q1 = predict(fit, newdata=pneumo)[1:3,]
q2 = predict(fit, newdata=pneumo[1:3,])
all.equal(q0, q1)   # Should be TRUE
all.equal(q1, q2)   # Should be TRUE
```

`model.matrixvlm` *Construct the Design Matrix of a VLM Object*

Description

Creates a design matrix. Two types can be returned: a large one (class `"vlm"` or one that inherits from this such as `"vglm"`) or a small one (such as returned if it were of class `"lm"`).

Usage

```
model.matrixvlm(object, type=c("vlm","lm"), ...)
```

Arguments

`object`	an object of a class that inherits from the *vector linear model* (VLM).
`type`	Type of design matrix returned. The first is the default.
`...`	further arguments passed to or from other methods. These include `data` (which is a data frame created with `model.framevlm`), `contrasts.arg`, and `xlev`. See `model.matrix` for more information.

Details

This function creates a design matrix from `object`. This can be a small LM object or a big VLM object (default). The latter is constructed from the former and the constraint matrices.

This code implements *smart prediction* (see `smartpred`).

Value

The design matrix for a regression model with the specified formula and data.

References

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

Chambers, J. M. (1992) *Data for models.* Chapter 3 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

See Also

model.matrix, model.framevlm, predict.vglm, smartpred.

Examples

```
# Illustrates smart prediction
data(pneumo)
pneumo = transform(pneumo, let=log(exposure.time))
fit = vglm(cbind(normal,mild, severe) ~ poly(c(scale(let)), 2),
           fam=multinomial,
           data=pneumo, trace=TRUE, x=FALSE)
class(fit)
fit@x
model.matrix(fit)

Check1 = model.matrix(fit, type="lm")[1:3,]
Check1
Check2 = model.matrix(fit, data=pneumo[1:3,], type="lm")
Check2
all.equal(c(Check1), c(Check2))

q0 = predict(fit)[1:3,]
q1 = predict(fit, newdata=pneumo)[1:3,]
q2 = predict(fit, newdata=pneumo[1:3,])
all.equal(q0, q1)  # Should be TRUE
all.equal(q1, q2)  # Should be TRUE
```

---

morgenstern *Morgenstern's Bivariate Distribution Family Function*

Description

Estimate the association parameter of Morgenstern's bivariate distribution using maximum likelihood estimation.

Usage

```
morgenstern(lapar="rhobit", earg=list(), iapar=NULL, tola0=0.01,
            method.init=1)
```

Arguments

lapar	Link function applied to the association parameter α, which lies between -1 and 1. See Links for more choices.
earg	List. Extra argument for the link. See earg in Links for general information.
iapar	Numeric. Optional initial value for α. By default, an initial value is chosen internally. If a convergence failure occurs try assigning a different value. Assigning a value will override the argument method.init.

`tola0` Positive numeric. If the estimate of α has an absolute value less than this then it is replaced by this value. This is an attempt to fix a numerical problem when the estimate is too close to zero.

`method.init` An integer with value `1` or `2` which specifies the initialization method. If failure to converge occurs try the other value, or else specify a value for `ia`.

Details

The cumulative distribution function is

$$P(Y_1 \leq y_1, Y_2 \leq y_2) = e^{-y_1-y_2}(1 + \alpha[1 - e^{-y_1}][1 - e^{-y_2}]) + 1 - e^{-y_1} - e^{-y_2}$$

for α between -1 and 1. The support of the function is for $y_1 > 0$ and $y_2 > 0$. The marginal distributions are an exponential distribution with unit mean. When $\alpha = 0$ then the random variables are independent, and this causes some problems in the estimation process since the distribution no longer depends on the parameter.

A variant of Newton-Raphson is used, which only seems to work for an intercept model. It is a very good idea to set `trace=TRUE`.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Note

The response must be a two-column matrix. Currently, the fitted value is a matrix with two columns and values equal to 1. This is because each marginal distribution corresponds to a exponential distribution with unit mean.

This **VGAM** family function should be used with caution.

Author(s)

T. W. Yee

References

Castillo, E., Hadi, A. S., Balakrishnan, N. Sarabia, J. S. (2005) *Extreme Value and Related Models with Applications in Engineering and Science*, Hoboken, N.J.: Wiley-Interscience.

See Also

`fgm`, `gumbelIbiv`.

Examples

```
n = 1000
ymat = cbind(rexp(n), rexp(n))
## Not run: plot(ymat)
fit = vglm(ymat ~ 1, fam=morgenstern, trace=TRUE)
fit = vglm(ymat ~ 1, fam=morgenstern, trace=TRUE, crit="coef")
coef(fit, matrix=TRUE)
Coef(fit)
fitted(fit)[1:5,]
```

---

multinomial *Multinomial Logit Model*

Description

Fits a multinomial logit model to an unordered factor response.

Usage

```
multinomial(zero = NULL, parallel = FALSE, nointercept = NULL)
```

Arguments

In the following, the response Y is assumed to be a factor with unordered values $1, 2, \ldots, M+1$, so that M is the number of linear/additive predictors η_j.

`zero`	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,...,M}. The default value means none are modelled as intercept-only terms.
`parallel`	A logical, or formula specifying which terms have equal/unequal coefficients.
`nointercept`	An integer-valued vector specifying which linear/additive predictors have no intercepts. The values must be from the set {1,2,...,M}.

Details

The model can be written

$$\eta_j = \log(P[Y = j]/P[Y = M + 1])$$

where η_j is the jth linear/additive predictor. Here, $j = 1, \ldots, M$ and η_{M+1} is 0 by definition. That is, the last level of the factor, or last column of the response matrix, is taken as the reference level or baseline—this is for identifiability of the parameters.

In almost all the literature, the constraint matrices associated with this family of models are known. For example, setting `parallel=TRUE` will make all constraint matrices (except for the intercept) equal to a vector of M 1's. If the constraint matrices are unknown and to be estimated, then this can be achieved by fitting the model as a reduced-rank vector generalized linear model (RR-VGLM; see `rrvglm`). In particular, a multinomial logit model with unknown constraint matrices is known as a stereotype model (Anderson, 1984), and can be fitted with `rrvglm`.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Warning

The arguments zero and nointercept can be inputted with values that fail. For example, multinomial(zero=2, nointercept=1:3) means the second linear/additive predictor is identically zero, which will cause a failure.

Be careful about the use of other potentially contradictory constraints, e.g., multinomial(zero=2, parallel = TRUE ~ x3). If in doubt, apply constraints() to the fitted object to check.

No check is made to verify that the response is nominal.

Note

The response should be either a matrix of counts (with row sums that are all positive), or a factor. In both cases, the y slot returned by vglm/vgam/rrvglm is the matrix of counts.

The multinomial logit model is more appropriate for a nominal (unordered) factor response. For an ordinal (ordered) factor response, models such as those based on cumulative probabilities (see cumulative) are more suited.

multinomial is prone to numerical difficulties if the groups are separable and/or the fitted probabilities are close to 0 or 1. The fitted values returned are estimates of the probabilities $P[Y = j]$ for $j = 1, \ldots, M + 1$.

Here is an example of the usage of the parallel argument. If there are covariates x1, x2 and x3, then parallel = TRUE ~ x1 + x2 -1 and parallel = FALSE ~ x3 are equivalent. This would constrain the regression coefficients for x1 and x2 to be equal; those of the intercepts and x3 would be different.

In Example 4 below, a conditional logit model is fitted to a artificial data set that explores how cost and travel time affect people's decision about how to travel to work. Walking is the baseline group. The variable Cost.car is the difference between the cost of travel to work by car and walking, etc. The variable Durn.car is the difference between the travel duration/time to work by car and walking, etc. For other details about the xij argument see vglm.control and fill.

The multinom function in the **nnet** package uses the first level of the factor as baseline, whereas the last level of the factor is used here. Consequently the estimated regression coefficients differ.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

Agresti, A. (2002) *Categorical Data Analysis*, 2nd ed. New York: Wiley.

Simonoff, J. S. (2003) *Analyzing Categorical Data*, New York: Springer-Verlag.

Anderson, J. A. (1984) Regression and ordered categorical variables. *Journal of the Royal Statistical Society, Series B, Methodological*, **46**, 1–30.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

acat, cumulative, cratio, sratio, dirichlet, dirmultinomial, rrvglm, Multinomial, iris.

Examples

```
# Example 1: fit a multinomial logit model to Edgar Anderson's iris data
data(iris)
## Not run:
fit = vglm(Species ~ ., multinomial, iris)
coef(fit, matrix=TRUE)
## End(Not run)

# Example 2a: a simple example
y = t(rmultinom(10, size = 20, prob=c(0.1,0.2,0.8))) # Counts
fit = vglm(y ~ 1, multinomial)
fitted(fit)[1:4,]   # Proportions
fit@prior.weights # Not recommended for extraction of prior weights
weights(fit, type="prior", matrix=FALSE) # The better method
fit@y   # Sample proportions
constraints(fit)   # Constraint matrices

# Example 2b: Different input to Example 2a but same result
w = apply(y, 1, sum) # Prior weights
yprop = y / w    # Sample proportions
fitprop = vglm(yprop ~ 1, multinomial, weights=w)
fitted(fitprop)[1:4,]   # Proportions
weights(fitprop, type="prior", matrix=FALSE)
fitprop@y # Same as the input

# Example 3: Fit a rank-1 stereotype model
data(car.all)
fit = rrvglm(Country ~ Width + Height + HP, multinomial, car.all, Rank=1)
coef(fit)   # Contains the C matrix
constraints(fit)$HP     # The A matrix
coef(fit, matrix=TRUE)  # The B matrix
Coef(fit)@C             # The C matrix
ccoef(fit)              # Better to get the C matrix this way
Coef(fit)@A             # The A matrix
svd(coef(fit, matrix=TRUE)[-1,])$d    # This has rank 1; = C

# Example 4: The use of the xij argument (conditional logit model)
set.seed(111)
n = 100  # Number of people who travel to work
M = 3  # There are M+1 models of transport
```

```
ymat = matrix(0, n, M+1)
ymat[cbind(1:n, sample(x=M+1, size=n, replace=TRUE))] = 1
dimnames(ymat) = list(NULL, c("bus","train","car","walk"))
transport = data.frame(cost.bus=runif(n), cost.train=runif(n),
                       cost.car=runif(n), cost.walk=runif(n),
                       durn.bus=runif(n), durn.train=runif(n),
                       durn.car=runif(n), durn.walk=runif(n))
transport = round(transport, dig=2) # For convenience
transport = transform(transport,
                      Cost.bus   = cost.bus   - cost.walk,
                      Cost.car   = cost.car   - cost.walk,
                      Cost.train = cost.train - cost.walk,
                      Durn.bus   = durn.bus   - durn.walk,
                      Durn.car   = durn.car   - durn.walk,
                      Durn.train = durn.train - durn.walk)
fit = vglm(ymat ~ Cost.bus + Cost.train + Cost.car +
                  Durn.bus + Durn.train + Durn.car,
           fam = multinomial,
           xij = list(Cost ~ Cost.bus + Cost.train + Cost.car,
                      Durn ~ Durn.bus + Durn.train + Durn.car),
           data=transport)
model.matrix(fit, type="lm")[1:7,]   # LM model matrix
model.matrix(fit, type="vlm")[1:7,]  # Big VLM model matrix
coef(fit)
coef(fit, matrix=TRUE)
coef(fit, matrix=TRUE, compress=FALSE)
summary(fit)
```

`nakagami` *Nakagami Distribution Family Function*

Description

Estimation of the two parameters of the Nakagami distribution by maximum likelihood estimation.

Usage

```
nakagami(lshape = "loge", lscale = "loge",
         eshape=list(), escale=list(), ishape = NULL, iscale = 1)
```

Arguments

`lshape, lscale`
: Parameter link function applied to the *shape* and *scale* parameters. Log links ensure they are positive. See `Links` for more choices.

`eshape, escale`
: List. Extra argument for each of the links. See `earg` in `Links` for general information.

`ishape, iscale`

Optional initial values for the shape and scale parameters. For `ishape`, a `NULL` value means it is obtained in the `initialize` slot based on the value of `iscale`. For `iscale`, assigning a `NULL` means a value is obtained in the `initialize` slot, however, setting another numerical value is recommended if convergence fails or is too slow.

Details

The Nakagami distribution, which is useful for modelling wireless systems such as radio links, can be written

$$f(y) = 2(shape/scale)^{shape} y^{2\times shape-1} \exp(-shape \times y^2/scale)/\Gamma(shape)$$

for $y > 0$, $shape > 0$, $scale > 0$. The mean of Y is $\sqrt{scale/shape} \times \Gamma(shape + 0.5)/\Gamma(shape)$ and these are returned as the fitted values. By default, the linear/additive predictors are $\eta_1 = \log(shape)$ and $\eta_2 = \log(scale)$. Fisher scoring is implemented.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

The Nakagami distribution is also known as the Nakagami-*m* distribution, where $m = shape$ here. Special cases: $m = 0.5$ is a one-sided Gaussian distribution and $m = 1$ is a Rayleigh distribution. The second moment is $E(Y^2) = m$.

If Y has a Nakagami distribution with parameters *shape* and *scale* then Y^2 has a gamma distribution with shape parameter *shape* and scale parameter *scale/shape*.

Author(s)

T. W. Yee

References

Nakagami, M. (1960) The *m*-distribution: a general formula of intensity distribution of rapid fading, pp.3–36 in: *Statistical Methods in Radio Wave Propagation*. W. C. Hoffman, Ed., New York: Pergamon.

See Also

`rnaka`, `gamma2`, `rayleigh`.

Examples

```
n = 1000; shape = exp(0); Scale = exp(1)
y = sqrt(rgamma(n, shape=shape, scale=Scale/shape))
fit = vglm(y ~ 1, nakagami, trace=TRUE, crit="c")
y = rnaka(n, shape=shape, scale=Scale)
```

```
fit = vglm(y ~ 1, nakagami(iscale=3), trace=TRUE)
fitted(fit)[1:5]
mean(y)
coef(fit, matrix=TRUE)
(Cfit = Coef(fit))
## Not run:
hist(sy <- sort(y), prob=TRUE, main="", xlab="y", ylim=c(0,0.6))
lines(sy, dnaka(sy, shape=Cfit[1], scale=Cfit[2]), col="red")
## End(Not run)
```

Nakagami *Nakagami Distribution*

Description

Density, cumulative distribution function, quantile function and random generation for the Nakagami distribution.

Usage

```
dnaka(x, shape, scale=1)
pnaka(q, shape, scale=1)
qnaka(p, shape, scale=1, ...)
rnaka(n, shape, scale=1, Smallno=1.0e-6)
```

Arguments

x, q	vector of quantiles.
p	vector of probabilities.
n	number of observations. Must be a positive integer of length 1.
shape, scale	arguments for the parameters of the distribution. See nakagami for more details. For rnaka, arguments shape and scale must be of length 1.
Smallno	Numeric, a small value used by the rejection method for determining the upper limit of the distribution. That is, pnaka(U) > 1-Smallno where U is the upper limit.
...	Arguments that can be passed into uniroot.

Details

See nakagami for more details.

Value

dnaka gives the density, pnaka gives the cumulative distribution function, qnaka gives the quantile function, and rnaka generates random deviates.

Author(s)

T. W. Yee

See Also

nakagami.

Examples

```
## Not run:
x = seq(0, 3.2, len=200)
plot(x, dgamma(x, shape=1), type="n", col="black", ylab="",
     ylim=c(0,1.5), main="dnaka(x, shape)")
lines(x, dnaka(x, shape=1), col="red")
lines(x, dnaka(x, shape=2), col="blue")
lines(x, dnaka(x, shape=3), col="green")
legend(2, 1.0, col=c("red","blue","green"), lty=rep(1, len=3),
       legend=paste("shape =", c(1, 2, 3)))

plot(x, pnorm(x), type="n", col="black", ylab="",
     ylim=0:1, main="pnaka(x, shape)")
lines(x, pnaka(x, shape=1), col="red")
lines(x, pnaka(x, shape=2), col="blue")
lines(x, pnaka(x, shape=3), col="green")
legend(2, 0.6, col=c("red","blue","green"), lty=rep(1, len=3),
       legend=paste("shape =", c(1, 2, 3)))
## End(Not run)

probs = seq(0.1, 0.9, by=0.1)
pnaka(qnaka(p=probs, shape=2), shape=2) - probs  # Should be all 0
```

---

nbolf *Negative Binomial-Ordinal Link Function*

Description

Computes the negative binomial-ordinal transformation, including its inverse and the first two derivatives.

Usage

```
nbolf(theta, earg = stop("'earg' must be given"), inverse = FALSE,
      deriv = 0, short = TRUE, tag = FALSE)
```

Arguments

theta	Numeric or character. See below for further details.
earg	Extra argument for passing in additional information. This must be list with components cutpoint and k. Here, k is the k parameter associated with the negative binomial distribution; see negbinomial. The cutpoints should be non-negative integers. If nbolf() is used as the link function in cumulative then one should choose reverse=TRUE, parallel=TRUE, intercept.apply=TRUE.
inverse	Logical. If TRUE the inverse function is computed.
deriv	Order of the derivative. Integer with value 0, 1 or 2.
short	Used for labelling the blurb slot of a vglmff-class object.
tag	Used for labelling the linear/additive predictor in the initialize slot of a vglmff-class object. Contains a little more information if TRUE.

Details

The negative binomial-ordinal link function (NBOLF) can be applied to a parameter lying in the unit interval. Its purpose is to link cumulative probabilities associated with an ordinal response coming from an underlying negative binomial distribution.

The arguments short and tag are used only if theta is character.

See Links for general information about **VGAM** link functions.

Value

See Yee (2006) for details.

Warning

Prediction may not work on vglm or vgam etc. objects if this link function is used.

Note

Numerical values of theta too close to 0 or 1 or out of range result in large positive or negative values, or maybe 0 depending on the arguments. Although measures have been taken to handle cases where theta is too close to 1 or 0, numerical instabilities may still arise.

In terms of the threshold approach with cumulative probabilities for an ordinal response this link function corresponds to the negative binomial distribution (see negbinomial) that has been recorded as an ordinal response using known cutpoints.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2006) *Ordinal ordination with normalizing link functions for count data*, (submitted for publication).

See Also

Links, negbinomial, polf, golf, nbolf2, cumulative.

Examples

```
earg = list(cutpoint=2, k=1)
nbolf("p", earg=earg, short=FALSE)
nbolf("p", earg=earg, tag=TRUE)

p = seq(0.02, 0.98, by=0.01)
y = nbolf(p, earg=earg)
y. = nbolf(p, earg=earg, deriv=1)
max(abs(nbolf(y, earg=earg, inv=TRUE) - p)) # Should be 0

## Not run:
par(mfrow=c(2,1), las=1)
plot(p, y, type="l", col="blue", main="nbolf()")
abline(h=0, v=0.5, col="red", lty="dashed")

plot(p, y., type="l", col="blue",
     main="(Reciprocal of) first NBOLF derivative")
## End(Not run)

# Another example
nn = 1000
x2 = sort(runif(nn))
x3 = runif(nn)
mymu = exp( 3 + 1 * x2 - 2 * x3)
k = 4
y1 = rnbinom(nn, mu=mymu, size=k)
cutpoints = c(-Inf, 10, 20, Inf)
cuty = Cut(y1, breaks=cutpoints)
## Not run:
plot(x2, x3, col=cuty, pch=as.character(cuty))
## End(Not run)
table(cuty) / sum(table(cuty))
fit = vglm(cuty ~ x2 + x3, fam = cumulative(link="nbolf",
           reverse=TRUE, parallel=TRUE, intercept.apply=TRUE,
           mv=TRUE, earg=list(cutpoint=cutpoints[2:3], k=k)),
           trace=TRUE)
fit@y[1:5,]
fitted(fit)[1:5,]
predict(fit)[1:5,]
coef(fit)
coef(fit, matrix=TRUE)
constraints(fit)
fit@misc$earg
```

---

negbinomial *Negative Binomial Distribution Family Function*

Description

Maximum likelihood estimation of the two parameters of a negative binomial distribution.

Usage

```
negbinomial(lmu = "loge", lk = "loge",
            emu =list(), ek=list(),
            ik = NULL, cutoff = 0.995, Maxiter=5000,
            deviance.arg = FALSE, method.init=1,
            shrinkage.init=0.95, zero = -2)
```

Arguments

`lmu, lk`	Link functions applied to the μ and k parameters. See `Links` for more choices.
`emu, ek`	List. Extra argument for each of the links. See `earg` in `Links` for general information.
`ik`	Optional initial values for k. If failure to converge occurs try different values (and/or use `method.init`). For a S-column response, `ik` can be of length S. A value `NULL` means an initial value for each response is computed internally using a range of values. This argument is ignored if used within `cqo`; see the `iKvector` argument of `qrrvglm.control` instead.
`cutoff`	A numeric which is close to 1 but never exactly 1. Used to specify how many terms of the infinite series for computing the second diagonal element of the expected information matrix are actually used. The sum of the probabilites are added until they reach this value or more (but no more than `Maxiter` terms allowed). It is like specifying `p` in an imaginary function `qnegbin(p)`.
`Maxiter`	Integer. The maximum number of terms allowed when computing the second diagonal element of the expected information matrix. In theory, the value involves an infinite series. If this argument is too small then the value may be inaccurate.
`deviance.arg`	Logical. If `TRUE`, the deviance function is attached to the object. Under ordinary circumstances, it should be left alone because it really assumes the index parameter is at the maximum likelihood estimate. Consequently, one cannot use that criterion to minimize within the IRLS algorithm. It should be set `TRUE` only when used with `cqo` under the fast algorithm.
`method.init`	An integer with value `1` or `2` which specifies the initialization method for the μ parameter. If failure to converge occurs try another value and/or else specify a value for `shrinkage.init` and/or else specify a value for `ik`.
`shrinkage.init`	How much shrinkage is used when initializing μ. The value must be between 0 and 1 inclusive, and a value of 0 means the individual response values are used, and a value of 1 means the median or mean is used. This argument is used in conjunction with `method.init`.
`zero`	Integer valued vector, usually assigned -2 or 2 if used at all. Specifies which of the two linear/additive predictors are modelled as an intercept only. By default, the k parameter (after `lk` is applied) is modelled as a single unknown number that is estimated. It can be modelled as a function of the explanatory variables

by setting `zero=NULL`. A negative value means that the value is recycled, so setting -2 means all k are intercept-only.

Details

The negative binomial distribution can be motivated in several ways, e.g., as a Poisson distribution with a mean that is gamma distributed. There are several common parametrizations of the negative binomial distribution. The one used here uses the mean μ and an *index* parameter k, both which are positive. Specifically, the density of a random variable Y is

$$f(y;\mu,k) = \binom{y+k-1}{y} \left(\frac{\mu}{\mu+k}\right)^y \left(\frac{k}{k+\mu}\right)^k$$

where $y = 0, 1, 2, \ldots$, and $\mu > 0$ and $k > 0$. Note that the dispersion parameter is $1/k$, so that as k approaches infinity the negative binomial distribution approaches a Poisson distribution. The response has variance $Var(Y) = \mu + \mu^2/k$. When fitted, the `fitted.values` slot of the object contains the estimated value of the μ parameter, i.e., of the mean $E(Y)$.

The negative binomial distribution can be coerced into the classical GLM framework, with one of the parameters being of interest and the other treated as a nuisance/scale parameter (and implemented in the MASS library). This **VGAM** family function `negbinomial` treats both parameters on the same footing, and estimates them both by full maximum likelihood estimation.

The parameters μ and k are independent (diagonal expected information matrix), and the confidence region for k is extremely skewed so that its standard error is often of no practical use. The parameter $1/k$ has been used as a measure of aggregation.

This **VGAM** function handles *multivariate* responses, so that a matrix can be used as the response. The number of columns is the number of species, say, and setting `zero=-2` means that *all* species have a k equalling a (different) intercept only.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Warning

The Poisson model corresponds to k equalling infinity. If the data is Poisson or close to Poisson, numerical problems will occur. Possibly choosing a log-log link may help in such cases, otherwise use `poissonff`.

This function is fragile; the maximum likelihood estimate of the index parameter is fraught (see Lawless, 1987). In general, the `quasipoissonff` is more robust than this function. Assigning values to the `ik` argument may lead to a local solution, and smaller values are preferred over large values when using this argument.

Yet to do: write a family function which uses the methods of moments estimator for k.

Note

This function can be used by the fast algorithm in `cqo`, however, setting `EqualTolerances=TRUE` and `ITolerances=FALSE` is recommended.

In the first example below (Bliss and Fisher, 1953), from each of 6 McIntosh apple trees in an orchard that had been sprayed, 25 leaves were randomly selected. On each of the leaves, the number of adult female European red mites were counted.

Author(s)

Thomas W. Yee

References

Lawless, J. F. (1987) Negative binomial and mixed Poisson regression. *The Canadian Journal of Statistics* **15**, 209–225.

Bliss, C. and Fisher, R. A. (1953) Fitting the negative binomial distribution to biological data. *Biometrics* **9**, 174–200.

See Also

quasipoissonff, poissonff, cao, cqo, posnegbinomial, rnbinom, nbolf.

Examples

```
y = 0:7  # Example 1: apple tree data
w = c(70, 38, 17, 10, 9, 3, 2, 1)
fit = vglm(y ~ 1, negbinomial, weights=w)
summary(fit)
coef(fit, matrix=TRUE)
Coef(fit)

## Not run:
n = 500  # Example 2: simulated data
x = runif(n)
y1 = rnbinom(n, mu=exp(3+x), size=exp(1)) # k is size
y2 = rnbinom(n, mu=exp(2-x), size=exp(0))
fit = vglm(cbind(y1,y2) ~ x, negbinomial, tra=TRUE) # multivariate response
coef(fit, matrix=TRUE)
## End(Not run)
```

---

normal1 *Univariate normal distribution*

Description

Maximum likelihood estimation of the two parameters of a univariate normal distribution.

Usage

```
normal1(lmean="identity", lsd="loge",
        emean=list(), esd=list(), zero=NULL)
```

Arguments

lmean	Link function applied to the mean. See Links for more choices.
lsd	Parameter link function applied to the standard deviation. See Links for more choices. Being a positive quantity, a log link is the default.
emean, esd	List. Extra argument for the links. See earg in Links for general information.
zero	An integer vector, containing the value 1 or 2. If so, the mean or standard deviation respectively are modelled as an intercept only. Usually, setting zero=2 will be used, if used at all. The default value NULL means both linear/additive predictors are modelled as functions of the explanatory variables.

Details

By default, the mean is the first linear/additive predictor and the log of the standard deviation is the second linear/additive predictor. The Fisher information matrix is diagonal.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

The response should be univariate. Multivariate responses are more generally handled using gaussianff, however this only handles the mean and the variance-covariance matrices are assumed known.

Author(s)

T. W. Yee

References

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

See Also

gaussianff, posnormal1, tobit, cnormal1, dcnormal1, studentt.

Examples

```
n = 200
x = rnorm(n)
y = rnorm(n, mean=1-3*x, sd=exp(1+0.2*x))
fit = vglm(y ~ x, normal1)
coef(fit, matrix=TRUE)

# Generate a random sample from a N(mu=theta, sigma=theta)
# distribution with theta=10. Then estimate theta.
theta = 10
```

```
y = rnorm(100, m=theta, sd=theta)
fit = vglm(y ~ 1, normal1(lsd="identity"),
           constraints=list("(Intercept)"=rbind(1,1)))
coef(fit, matrix=TRUE)
```

`notdocumentedyet` *Undocumented and Internally Used Functions and Classes*

Description

Those currently undocumented and internally used functions are aliased to this help file. Ditto for some classes.

Details

In the **VGAM** package there are currently many objects/methods/classes which are currently internal and/or undocumented. The help file suppresses the warnings when the package is 'CHECK'ed.

Value

Each objects/methods/classes may or may not have its own individual value. These will be documented over time.

Author(s)

T. W. Yee

`nzc` *Chinese Population in New Zealand 1867–2001*

Description

The Chinese population in New Zealand from 1867 to 2001, along with the whole of the New Zealand population.

Usage

```
data(nzc)
```

Format

A data frame with 26 observations on the following 4 variables.

`year` Year.

`male` Number of Chinese males.

`female` Number of Chinese females.

`nz` Total number in the New Zealand population.

Details

The NZ total for the years 1867 and 1871 exclude the Maori population. The second value of 4583 looks erroneous, as seen by the plot below.

References

Page 6 of *Aliens At My Table: Asians as New Zealanders see them* by M. Ip and N. Murphy, (2005), Penguin.

Examples

```
## Not run:
data(nzc)
attach(nzc)
plot(year, female/(male+female), type="b", ylab="Proportion",
     main="Proportion of Chinese that are female",
     col="blue", las=1)
abline(h=0.5, lty="dashed")

plot(year, 100*(male+female)/nz, type="b", ylab="Percent",
     ylim=c(0, max(100*(male+female)/nz)), col="blue",
     main="Percent of New Zealanders that are Chinese", las=1)
abline(h=0, lty="dashed")

detach(nzc)
## End(Not run)
```

---

ordpoisson *Ordinal Poisson Family Function*

Description

Fits a Poisson regression where the response is ordinal (the Poisson counts are grouped between known cutpoints).

Usage

```
ordpoisson(cutpoints, countdata=FALSE, NOS=NULL,
           Levels=NULL, init.mu=NULL, parallel=FALSE,
           zero=NULL, link="loge", earg = list())
```

Arguments

`cutpoints`	Numeric. The cutpoints, K_l. These must be non-negative integers. `Inf` values may be included. See below for further details.

`countdata` Logical. Is the response (LHS of formula) in count-data format? If not then the response is a matrix or vector with values $1, 2, \ldots,$ `L`, say, where `L` is the number of levels. Such input can be generated with `cut` with argument `labels = FALSE`. If `countdata = TRUE` then the response is expected to be in the same format as `fit@y` where `fit` is a fitted model with `ordpoisson` as the **VGAM** family function. That is, the response is matrix of counts with `L` columns (if `NOS=1`).

`NOS` Integer. The number of species, or more generally, the number of response random variates. This argument must be specified when `countdata=TRUE`. Usually `NOS=1`.

`Levels` Integer vector, recycled to length `NOS` if necessary. The number of levels for each response random variate. This argument should agree with `cutpoints`. This argument must be specified when `countdata=TRUE`.

`init.mu` Numeric. Initial values for the means of the Poisson regressions. Recycled to length `NOS` if necessary. Use this argument if the default initial values fail (the default is to compute an initial value internally).

`parallel, zero, link, earg` See `poissonff`.

Details

This **VGAM** family function uses maximum likelihood estimation (Fisher scoring) to fit a Poisson regression to each column of a matrix response. The data, however, is ordinal, and is obtained from known integer cutpoints. Here, $l = 1, \ldots, L$ where L ($L \geq 2$) is the number of levels. In more detail, let $Y^* = l$ if $K_{l-1} < Y \leq K_l$ where the K_l are the cutpoints. We have $K_0 = -\infty$ and $K_L = \infty$. The response for this family function corresponds to Y^* but we are really interested in the Poisson regression of Y.

If `NOS=1` then the argument `cutpoints` is a vector $(K_1, K_2, \ldots, K_L)$ where the last value (`Inf`) is optional. If `NOS>1` then the vector should have `NOS-1 Inf` values separating the cutpoints. For example, if there are `NOS=3` responses, then something like `ordpoisson(cut = c(0, 5, 10, Inf, 20, 30, Inf, 0, 10, 40, Inf))` is valid.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Warning

The input requires care as little to no checking is done. If `fit` is the fitted object, have a look at `fit@extra` and `fit@y` to check.

Note

Sometimes there are no observations between two cutpoints. If so, the arguments `Levels` and `NOS` need to be specified too. See below for an example.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2006) *Ordinal ordination with normalizing link functions for count data*, (submitted for publication).

See Also

poissonff, polf, ordered.

Examples

```
# Example 1
set.seed(123)
x2 = runif(n <- 1000); x3 = runif(n)
mymu = exp(3 - 1 * x2 + 2 * x3)
y1 = rpois(n, lambda=mymu)
cutpts = c(-Inf, 20, 30, Inf)
fcutpts = cutpts[is.finite(cutpts)]  # finite cutpoints
ystar = cut(y1, breaks=cutpts, labels=FALSE)
## Not run:
plot(x2, x3, col=ystar, pch=as.character(ystar))
## End(Not run)
table(ystar) / sum(table(ystar))
fit = vglm(ystar ~ x2 + x3, fam = ordpoisson(cutpoi=fcutpts))
fit@y[1:5,]  # This can be input if countdata=TRUE
fitted(fit)[1:5,]
predict(fit)[1:5,]
coef(fit, matrix=TRUE)
fit@extra

# Example 2: multivariate and there are no obsns between some cutpoints
cutpts2 = c(-Inf, 0, 9, 10, 20, 70, 200, 201, Inf)
fcutpts2 = cutpts2[is.finite(cutpts2)]  # finite cutpoints
y2 = rpois(n, lambda=mymu)   # Same model as y1
ystar2 = cut(y2, breaks=cutpts2, labels=FALSE)
table(ystar2) / sum(table(ystar2))
fit = vglm(cbind(ystar,ystar2) ~ x2 + x3, fam =
           ordpoisson(cutpoi=c(fcutpts,Inf,fcutpts2,Inf),
                      Levels=c(length(fcutpts)+1,length(fcutpts2)+1),
                      parallel=TRUE), trace=TRUE)
coef(fit, matrix=TRUE)
fit@extra
constraints(fit)
summary(fit@y)  # Some columns have all zeros
```

oxtemp *Oxford Temperature Data*

Description

Annual maximum temperatures collected at Oxford, UK.

Usage

```
data(oxtemp)
```

Format

A data frame with 80 observations on the following 2 variables.

maxtemp Annual maximum temperatures (in degrees Fahrenheit).

year The values 1901 to 1980.

Details

The data were collected from 1901 to 1980.

Source

Unknown.

Examples

```
data(oxtemp)
fit = vglm(maxtemp ~ 1, egev, trace=TRUE, data=oxtemp)
```

paralogistic *Paralogistic Distribution Family Function*

Description

Maximum likelihood estimation of the 2-parameter paralogistic distribution.

Usage

```
paralogistic(link.a = "loge", link.scale = "loge",
             earg.a=list(), earg.scale=list(),
             init.a = 1, init.scale = NULL, zero = NULL)
```

Arguments

`link.a, link.scale`
: Parameter link functions applied to the (positive) shape parameter `a` and (positive) scale parameter `scale`. See `Links` for more choices.

`earg.a, earg.scale`
: List. Extra argument for each of the links. See `earg` in `Links` for general information.

`init.a, init.scale`
: Optional initial values for `a` and `scale`.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. Here, the values must be from the set {1,2} which correspond to `a`, `scale`, respectively.

Details

The 2-parameter paralogistic distribution is the 4-parameter generalized beta II distribution with shape parameter $p = 1$ and $a = q$. It is the 3-parameter Singh-Maddala distribution with $a = q$. More details can be found in Kleiber and Kotz (2003).

The 2-parameter paralogistic has density

$$f(y) = a^2 y^{a-1}/[b^a\{1 + (y/b)^a\}^{1+a}]$$

for $a > 0$, $b > 0$, $y > 0$. Here, b is the scale parameter `scale`, and a is the shape parameter. The mean is

$$E(Y) = b\,\Gamma(1 + 1/a)\,\Gamma(a - 1/a)/\Gamma(a)$$

provided $a > 1$.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

If the self-starting initial values fail, try experimenting with the initial value arguments, especially those whose default value is not `NULL`.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

See Also

`Paralogistic`, `genbetaII`, `betaII`, `dagum`, `fisk`, `invlomax`, `lomax`, `invparalogistic`.

Examples

```
y = rparalogistic(n=3000, 4, 6)
fit = vglm(y ~ 1, paralogistic, trace=TRUE)
fit = vglm(y ~ 1, paralogistic(init.a=2.3, init.sc=5),
           trace=TRUE, crit="c")
coef(fit, mat=TRUE)
Coef(fit)
summary(fit)
```

`pareto1` *Pareto and Truncated Pareto Distribution Family Functions*

Description

Estimates one of the parameters of the Pareto(I) distribution by maximum likelihood estimation. Also includes the upper truncated Pareto(I) distribution.

Usage

```
pareto1(lshape = "loge", earg=list(), location=NULL)
tpareto1(lower, upper, lshape = "loge", earg=list(), ishape=NULL,
         method.init=1)
```

Arguments

`lshape`	Parameter link function applied to the parameter k. See `Links` for more choices. A log link is the default because k is positive.
`earg`	List. Extra argument for the link. See `earg` in `Links` for general information.
`lower, upper`	Numeric. Lower and upper limits for the truncated Pareto distribution. Each must be positive and of length 1. They are called α and U below.
`ishape`	Numeric. Optional initial value for the shape parameter. A `NULL` means a value is obtained internally. If failure to converge occurs try specifying a value, e.g., 1 or 2.
`location`	Numeric. The parameter α below. If the user inputs a number then it is assumed known with this value. The default means it is estimated by maximum likelihood estimation, which means `min(y)` where `y` is the response vector.
`method.init`	An integer with value `1` or `2` which specifies the initialization method. If failure to converge occurs try the other value, or else specify a value for `ishape`.

Details

A random variable Y has a Pareto distribution if

$$P[Y > y] = C/y^k$$

for some positive k and C. This model is important in many applications due to the power law probability tail, especially for large values of y.

The Pareto distribution, which is used a lot in economics, has a probability density function that can be written

$$f(y) = k\alpha^k / y^{k+1}$$

for $0 < \alpha < y$ and $k > 0$. The α is known as the location parameter, and k is known as the shape parameter. The mean of Y is $\alpha k/(k-1)$ provided $k > 1$. Its variance is $\alpha^2 k/((k-1)^2(k-2))$ provided $k > 2$.

The upper truncated Pareto distribution has a probability density function that can be written

$$f(y) = k\alpha^k / [y^{k+1}(1-(\alpha/U)^k)]$$

for $0 < \alpha < y < U < \infty$ and $k > 0$. Possibly, better names for k are the *index* and *tail* parameters. Here, α and U are known. The mean of Y is $k\alpha^k(U^{1-k} - \alpha^{1-k})/[(1-k)(1-(\alpha/U)^k)]$.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Warning

The usual or unbounded Pareto distribution has two parameters (called α and k here) but the family function `pareto1` estimates only k using iteratively reweighted least squares. The MLE of the α parameter lies on the boundary and is `min(y)` where `y` is the response. Consequently, using the default argument values, the standard errors are incorrect when one does a `summary` on the fitted object. If the user inputs a value for `alpha` then it is assumed known with this value and then `summary` on the fitted object should be correct. Numerical problems may occur for small k, e.g., $k < 1$.

Note

Outside of economics, the Pareto distribution is known as the Bradford distribution.

For `pareto1`, if the estimate of k is less than or equal to unity then the fitted values will be `NA`s. Also, `pareto1` fits the Pareto(I) distribution. See `paretoIV` for the more general Pareto(IV/III/II) distributions, but there is a slight change in notation: $s = k$ and $b = \alpha$.

In some applications the Pareto law is truncated by a natural upper bound on the probability tail. The upper truncated Pareto distribution has three parameters (called α, U and k here) but the family function `tpareto` estimates only k. With known lower and upper limits, the ML estimator of k has the usual properties of MLEs. Aban (2006) discusses other inferential details.

Author(s)

T. W. Yee

References

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

Aban, I. B., Meerschaert, M. M. and Panorska, A. K. (2006) Parameter estimation for the truncated Pareto distribution, *Journal of the American Statistical Association*, **101**(473), 270–277.

See Also

Pareto, Tpareto, paretoIV, gpd.

Examples

```
alpha = 2; k = exp(3)
y = rpareto(n=1000, location=alpha, shape=k)
fit = vglm(y ~ 1, pareto1, trace=TRUE)
fit@extra   # The estimate of alpha is here
fitted(fit)[1:5]
mean(y)
coef(fit, matrix=TRUE)
summary(fit)     # Standard errors are incorrect!!

# Here, alpha is assumed known
fit2 = vglm(y ~ 1, pareto1(location=alpha), trace=TRUE, crit="c")
fit2@extra   # alpha stored here
fitted(fit2)[1:5]
mean(y)
coef(fit2, matrix=TRUE)
summary(fit2)    # Standard errors are ok

# Upper truncated Pareto distribution
lower = 2; upper = 8; k = exp(2)
y = rtpareto(n=100, lower=lower, upper=upper, shape=k)
fit3 = vglm(y ~ 1, tpareto1(lower, upper), trace=TRUE, cri="c")
coef(fit3, matrix=TRUE)
c(fit3@misc$lower, fit3@misc$upper)
```

---

paretoIV *Pareto(IV/III/II) Distribution Family Functions*

Description

Estimates three of the parameters of the Pareto(IV) distribution by maximum likelihood estimation. Some special cases of this distribution are also handled.

Usage

```
paretoIV(location=0, lscale="loge", linequality="loge", lshape="loge",
         escale=list(), einequality=list(), eshape=list(),
         iscale=1, iinequality=1, ishape=NULL, method.init=1)
paretoIII(location=0, lscale="loge", linequality="loge",
          escale=list(), einequality=list(),
          iscale=NULL, iinequality=NULL)
paretoII(location=0, lscale="loge", lshape="loge",
         escale=list(), eshape=list(),
         iscale=NULL, ishape=NULL)
```

Arguments

`location` Location parameter, called a below. It is assumed known.

`lscale, linequality, lshape`
Parameter link functions for the scale parameter (called b below), inequality parameter (called g below), and shape parameter (called s below). See `Links` for more choices. A log link is the default for all because all these parameters are positive.

`escale, einequality, eshape`
List. Extra argument for each of the links. See `earg` in `Links` for general information.

`iscale, iinequality, ishape`
Initial values for the parameters. A `NULL` value means that it is obtained internally. If convergence failure occurs, use these arguments to input some alternative initial values.

`method.init` Method of initialization for the shape parameter. Currently only values 1 and 2 are available. Try the other value if convergence failure occurs.

Details

The Pareto(IV) distribution, which is used in actuarial science, economics, finance and telecommunications, has a cumulative distribution function that can be written

$$F(y) = 1 - [1 + ((y-a)/b)^{1/g}]^{-s}$$

for $y > a$, $b > 0$, $g > 0$ and $s > 0$. The a is called the *location* parameter, b the *scale* parameter, g the *inequality* parameter, and s the *shape* parameter.

The location parameter is assumed known otherwise the Pareto(IV) distribution will not be a regular family. This assumption is not too restrictive in modelling because in typical applications this parameter is known, e.g., in insurance and reinsurance it is pre-defined by a contract and can be represented as a deductible or a retention level.

The inequality parameter is so-called because of its interpretation in the economics context. If we choose a unit shape parameter value and a zero location parameter value then the inequality parameter is the Gini index of inequality, provided $g \leq 1$.

The fitted values are currently `NA` because I haven't worked out what the mean of Y is yet.

There are a number of special cases of the Pareto(IV) distribution. These include the Pareto(I), Pareto(II), Pareto(III), and Burr family of distributions. Denoting $PIV(a, b, g, s)$ as the Pareto(IV) distribution, the Burr distribution $Burr(b, g, s)$ is $PIV(a = 0, b, 1/g, s)$, the Pareto(III) distribution $PIII(a, b, g)$ is $PIV(a, b, g, s = 1)$, the Pareto(II) distribution $PII(a, b, s)$ is $PIV(a, b, g = 1, s)$, and the Pareto(I) distribution $PI(b, s)$ is $PIV(b, b, g = 1, s)$. Thus the Burr distribution can be fitted using the `nloge` link function and using the default `location=0` argument. The Pareto(I) distribution can be fitted using `pareto1` but there is a slight change in notation: $s = k$ and $b = \alpha$.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

The `extra` slot of the fitted object has a component called `"location"` which stores the location parameter value(s).

Author(s)

T. W. Yee

References

Brazauskas, V. (2003) Information matrix for Pareto(IV), Burr, and related distributions. *Comm. Statist. Theory and Methods* **32**, 315–325.

Arnold, B. C. (1983) *Pareto Distributions*. Fairland, Maryland: International Cooperative Publishing House.

See Also

`ParetoIV`, `pareto1`, `gpd`.

Examples

```
y = rparetoIV(n <- 2000, scale=exp(1), ineq=exp(-0.3), shape=exp(1))
## Not run: par(mfrow=c(2,1)); hist(y); hist(log(y));
fit = vglm(y ~ 1, paretoIV, trace=TRUE)
coef(fit, matrix=TRUE)
Coef(fit)
summary(fit)
```

---

persp.qrrvglm *Perspective plot for QRR-VGLMs*

Description

Produces a perspective plot for a CQO model (QRR-VGLM). It is only applicable for rank-1 or rank-2 models with argument `Norrr = ~ 1`.

Usage

```
persp.qrrvglm(x, varlvI = FALSE, reference = NULL, plot.it = TRUE,
              xlim=NULL, ylim=NULL, zlim=NULL,
              gridlength=if(Rank==1) 301 else c(51,51),
              whichSpecies = NULL,
              xlab = if(Rank==1) "Latent Variable" else "Latent Variable 1",
              ylab = if(Rank==1) "Expected Value" else "Latent Variable 2",
              zlab = "Expected value", labelSpecies = FALSE,
              stretch = 1.05, main="", ticktype = "detailed",
              col = if(Rank==1) par()$col else "white", add1 = FALSE, ...)
```

Arguments

x	Object of class "qrrvglm", i.e., a constrained quadratic ordination (CQO) object.
varlvI	Logical that is fed into Coef.qrrvglm.
reference	Integer or character that is fed into Coef.qrrvglm.
plot.it	Logical. Plot it?
xlim, ylim	Limits of the x- and y-axis. Both are numeric of length 2. See par.
zlim	Limits of the z-axis. Numeric of length 2. Ignored if rank is 1. See par.
gridlength	Numeric. The fitted values are evaluated on a grid, and this argument regulates the fineness of the grid. If Rank=2 then the argument is recycled to length 2, and the two numbers are the number of grid points on the x- and y-axes respectively.
whichSpecies	Numeric or character vector. Indicates which species are to be plotted. The default is to plot all of them. If numeric, it should contain values in the set {1,2,...,S} where S is the number of species.
xlab, ylab	Character caption for the x-axis and y-axis. By default, a suitable caption is found. See the xlab argument in plot or title.
zlab	Character caption for the z-axis. Used only if Rank=2. By default, a suitable caption is found. See the xlab argument in plot or title.
labelSpecies	Logical. Whether the species should be labelled with their names. Used for Rank=1 only. The position of the label is just above the species' maximum.
stretch	Numeric. A value slightly more than 1, this argument adjusts the height of the y-axis. Used for Rank=1 only.
main	Character, giving the title of the plot. See the main argument in plot or title.
ticktype	Tick type. Used only if Rank=2. See persp for more information.
col	Color. See persp for more information.
add1	Logical. Add to an existing plot? Used only for rank-1 models.
...	Arguments passed into persp. Useful arguments here include theta and phi, which control the position of the eye.

Details

For a rank-1 model, a perspective plot is similar to lvplot.qrrvglm but plots the curves along a fine grid and there is no rugplot to show the site scores.

For a rank-2 model, a perspective plot has the first latent variable as the x-axis, the second latent variable as the y-axis, and the expected value (fitted value) as the z-axis. The result of a CQO is that each species has a response surface with elliptical contours. This function will, at each grid point, work out the maximum fitted value over all the species. The resulting response surface is plotted. Thus rare species will be obscured and abundant species will dominate the plot. To view rare species, use the whichSpecies argument to select a subset of the species.

A perspective plot will be performed if Norrr = ~ 1, and Rank = 1 or 2. Also, all the tolerance matrices of those species to be plotted must be positive-definite.

Value

For a rank-2 model, a list with the following components.

`fitted` A $(G_1 \times G_2)$ by M matrix of fitted values on the grid. Here, G_1 and G_2 are the two values of `gridlength`.

`lv1grid, lv2grid` The grid points for the x-axis and y-axis.

`maxfitted` A G_1 by G_2 matrix of maximum of the fitted values over all species. These are the values that are plotted on the z-axis.

For a rank-1 model, the components `lv2grid` and `maxfitted` are `NULL`.

Note

Yee (2004) does not refer to perspective plots. Instead, contour plots via `lvplot.qrrvglm` are used.

For rank-1 models, a similar function to this one is `lvplot.qrrvglm`. It plots the fitted values at the actual site score values rather than on a fine grid here. The result has the advantage that the user sees the curves as a direct result from a model fitted to data whereas here, it is easy to think that the smooth bell-shaped curves are the truth because the data is more of a distance away.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2004) A new technique for maximum-likelihood canonical Gaussian ordination. *Ecological Monographs*, **74**, 685–701.

See Also

`persp`, `cqo`, `Coef.qrrvglm`, `lvplot.qrrvglm`, `par`, `title`.

Examples

```
## Not run:
data(hspider)
hspider[,1:6] = scale(hspider[,1:6]) # Good idea when ITolerances = TRUE
set.seed(111)
r1 = cqo(cbind(Alopacce, Alopcune, Alopfabr, Arctlute, Arctperi,
               Auloalbi, Pardmont, Pardnigr, Pardpull, Trocterr) ~
         WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + ReflLux,
         poissonff, hspider, trace = FALSE, ITolerances = TRUE)
set.seed(111)  # r2 below is an ill-conditioned model
r2 = cqo(cbind(Alopacce, Alopcune, Alopfabr, Arctlute, Arctperi,
               Auloalbi, Pardmont, Pardnigr, Pardpull, Trocterr) ~
         WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + ReflLux,
         isdlv = c(2.4,1.0), Muxfactor = 3.0, trace = FALSE,
         poissonff, hspider, Rank = 2, EqualTolerances = TRUE)
```

```
sort(r1@misc$deviance.Bestof)  # A history of the fits
sort(r2@misc$deviance.Bestof)  # A history of the fits
if(deviance(r2) > 857) stop("suboptimal fit obtained")

persp(r1, xlim=c(-6,5), col=1:4, label=TRUE)

# Involves all species
persp(r2, xlim=c(-6,5), ylim=c(-4,5), theta=10, phi=20, zlim=c(0,220))
# Omit the two dominant species to see what's behind them
persp(r2, xlim=c(-6,5), ylim=c(-4,5), theta=10, phi=20, zlim=c(0,220),
      which=(1:10)[-c(8,10)]) # Use zlim to retain the original z-scale
## End(Not run)
```

---

plotdeplot.lmscreg *Density Plot for LMS Quantile Regression*

Description

Plots a probability density function associated with a LMS quantile regression.

Usage

```
plotdeplot.lmscreg(answer, y.arg, add.arg = FALSE,
    xlab = "", ylab = "density",
    xlim = NULL, ylim = NULL,
    llty.arg = par()$lty, col.arg = par()$col,
    llwd.arg = par()$lwd, ...)
```

Arguments

`answer`	Output from functions of the form `deplot.???` where `???` is the name of the **VGAM** LMS family function, e.g., `lms.yjn`. See below for details.
`y.arg`	Numerical vector. The values of the response variable at which to evaluate the density. This should be a grid that is fine enough to ensure the plotted curves are smooth.
`add.arg`	Logical. Add the density to an existing plot?
`xlab, ylab`	Caption for the x- and y-axes. See `par`.
`xlim, ylim`	Limits of the x- and y-axes. See `par`.
`llty.arg`	Line type. See the `lty` argument of `par`.
`col.arg`	Line color. See the `col` argument of `par`.
`llwd.arg`	Line width. See the `lwd` argument of `par`.
`...`	Arguments passed into the `plot` function when setting up the entire plot. Useful arguments here include `main` and `las`.

Details

The above graphical parameters offer some flexibility when plotting the quantiles.

Value

The list `answer`, which has components

`newdata`	The argument `newdata` above from the argument list of `deplot.lmscreg`, or a one-row data frame constructed out of the `x0` argument.
`y`	The argument `y.arg` above.
`density`	Vector of the density function values evaluated at `y.arg`.

Note

While the graphical arguments of this function are useful to the user, this function should not be called directly.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2004) Quantile regression via vector generalized additive models. *Statistics in Medicine*, **23**, 2295–2315.

Documentation accompanying the **VGAM** package at `http://www.stat.auckland.ac.nz/~yee` contains further information and examples.

See Also

`deplot.lmscreg`.

Examples

```
data(bminz)
fit = vgam(BMI ~ s(age, df=c(4,2)), fam=lms.bcn(zero=1), data=bminz)
## Not run:
y = seq(15, 43, by=0.25)
deplot(fit, x0=20, y=y, xlab="BMI", col="green", llwd=2,
       main="BMI distribution at ages 20 (green), 40 (blue), 60 (red)")
deplot(fit, x0=40, y=y, add=TRUE, col="blue", llwd=2)
deplot(fit, x0=60, y=y, add=TRUE, col="red", llwd=2) -> a

names(a@post$deplot)
a@post$deplot$newdata
a@post$deplot$y[1:5]
a@post$deplot$density[1:5]
## End(Not run)
```

plotqrrvglm *Model Diagnostic Plots for QRR-VGLMs*

Description

The residuals of a QRR-VGLM are plotted for model diagnostic purposes.

Usage

```
plotqrrvglm(object,
            rtype = c("pearson", "response", "deviance", "working"),
            ask = FALSE,
            main = paste(Rtype, "residuals vs latent variable(s)"),
            xlab = "Latent Variable",
            ITolerances = object@control$EqualTolerances, ...)
```

Arguments

object	An object of class "qrrvglm".
rtype	Character string giving residual type. By default, the first one is chosen.
ask	Logical. If TRUE, the user is asked to hit the return key for the next plot.
main	Character string giving the title of the plot.
xlab	Character string giving the x-axis caption.
ITolerances	Logical. This argument is fed into Coef(object, ITolerances=ITolerances).
...	Other plotting arguments (see par).

Details

Plotting the residuals can be potentially very useful for checking that the model fit is adequate.

Value

The original object.

Note

An ordination plot of a QRR-VGLM can be obtained by lvplot.qrrvglm.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2004) A new technique for maximum-likelihood canonical Gaussian ordination. *Ecological Monographs*, **74**, 685–701.

See Also

lvplot.qrrvglm, cqo.

Examples

```
## Not run:
# QRR-VGLM on the hunting spiders data
# This is computationally expensive
data(hspider)
set.seed(111)  # This leads to the global solution
# hspider[,1:6]=scale(hspider[,1:6]) # Standardize the environmental variables
p1 = cqo(cbind(Alopacce, Alopcune, Alopfabr, Arctlute, Arctperi,
               Auloalbi, Pardlugu, Pardmont, Pardnigr, Pardpull,
               Trocterr, Zoraspin) ~
         WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + ReflLux,
         fam = quasipoissonff, data = hspider, Crow1positive=FALSE)
par(mfrow=c(3,4))
plot(p1, rtype="d", col="blue", pch=4, las=1)
## End(Not run)
```

---

plotqtplot.lmscreg *Quantile Plot for LMS Quantile Regression*

Description

Plots the quantiles associated with a LMS quantile regression.

Usage

```
plotqtplot.lmscreg(fitted.values, object, newdata = NULL,
    percentiles = object@misc$percentiles, lp = NULL,
    add.arg = FALSE, y = if (length(newdata)) FALSE else TRUE,
    spline.fit = FALSE, label = TRUE, size.label = 0.06,
    xlab = NULL, ylab = "",
    pch = par()$pch, pcex = par()$cex, pcol.arg = par()$col,
    xlim = NULL, ylim = NULL,
    llty.arg = par()$lty, lcol.arg = par()$col, llwd.arg = par()$lwd,
    tcol.arg = par()$col, tadj = 1, ...)
```

Arguments

fitted.values	Matrix of fitted values.
object	A **VGAM** quantile regression model, i.e., an object produced by modelling functions such as vglm and vgam with a family function beginning with "lms.", e.g., lms.yjn.
newdata	Data frame at which predictions are made. By default, the original data are used.

percentiles	Numerical vector with values between 0 and 100 that specify the percentiles (quantiles). The default is to use the percentiles when fitting the model. For example, the value 50 corresponds to the median.
lp	Length of percentiles.
add.arg	Logical. Add the quantiles to an existing plot?
y	Logical. Add the response as points to the plot?
spline.fit	Logical. Add a spline curve to the plot?
label	Logical. Add the percentiles (as text) to the plot?
size.label	Numeric. How much room to leave at the RHS for the label. It is in percent (of the range of the primary variable).
xlab	Caption for the x-axis. See par.
ylab	Caption for the x-axis. See par.
pch	Plotting character. See par.
pcex	Character expansion of the points. See par.
pcol.arg	Color of the points. See the col argument of par.
xlim	Limits of the x-axis. See par.
ylim	Limits of the y-axis. See par.
llty.arg	Line type. Line type. See the lty argument of par.
lcol.arg	Color of the lines. See the col argument of par.
llwd.arg	Line width. See the lwd argument of par.
tcol.arg	Color of the text (if label is TRUE). See the col argument of par.
tadj	Text justification. See the adj argument of par.
...	Arguments passed into the plot function when setting up the entire plot. Useful arguments here include main and las.

Details

The above graphical parameters offer some flexibility when plotting the quantiles.

Value

The matrix of fitted values.

Note

While the graphical arguments of this function are useful to the user, this function should not be called directly.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2004) Quantile regression via vector generalized additive models. *Statistics in Medicine*, **23**, 2295–2315.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

qtplot.lmscreg.

Examples

```
## Not run:
data(bminz)
fit = vgam(BMI ~ s(age, df=c(4,2)), fam=lms.bcn(zero=1), data=bminz)
qtplot(fit)
qtplot(fit, perc=c(25,50,75,95), lcol="blue", tcol="blue", llwd=2)
## End(Not run)
```

---

plotvgam *Default VGAM Plotting*

Description

Component functions of a vgam-class object can be plotted with plotvgam(). These are on the scale of the linear/additive predictor.

Usage

```
plotvgam(x, newdata = NULL, y = NULL, residuals = NULL,
         rugplot = TRUE, se = FALSE, scale = 0, raw = TRUE,
         offset.arg = 0, deriv.arg = 0, overlay = FALSE,
         type.residuals = c("deviance", "working", "pearson", "response"),
         plot.arg = TRUE, which.term = NULL, which.cf = NULL,
         control = plotvgam.control(...), ...)
```

Arguments

x	A fitted **VGAM** object, e.g., produced by vgam(), vglm(), or rrvglm().
newdata	Data frame. May be used to reconstruct the original data set.
y	Unused.
residuals	Logical. If TRUE, residuals are plotted. See type.residuals
rugplot	Logical. If TRUE, a rug plot is plotted at the foot of each plot. These values are jittered to expose ties.
se	Logical. If TRUE, approximate ± 2 pointwise standard error bands are included in the plot.

`scale` Numerical. By default, each plot will have its own y-axis scale. However, by specifying a value, each plot's y-axis scale will be at least `scale` wide.

`raw` Logical. If `TRUE`, the smooth functions are those obtained directly by the algorithm, and are plotted without having to premultiply with the constraint matrices. If `FALSE`, the smooth functions have been premultiply by the constraint matrices. The `raw` argument is directly fed into `predict.vgam()`.

`offset.arg` Numerical vector of length r. These are added to the component functions. Useful for separating out the functions when `overlay` is `TRUE`. If `overlay` is `TRUE` and there is one covariate, using the intercept values as the offsets can be a good idea.

`deriv.arg` Numerical. The order of the derivative. Should be assigned an small integer such as 0, 1, 2. Only applying to `s()` terms, it plots the derivative.

`overlay` Logical. If `TRUE`, component functions of the same covariate are overlaid on each other. The functions are centered, so `offset.arg` can be useful when `overlay` is `TRUE`.

`type.residuals` if `residuals` is `TRUE`, the first possible value of this vector, is used to specify the type of residual.

`plot.arg` Logical. If `FALSE`, no plot is produced.

`which.term` Character or integer vector containing all terms to be plotted, e.g., `which.term=c("s(age)", "s(height")` or `which.term=c(2,5,9)`. By default, all are plotted.

`which.cf` An integer-valued vector specifying which linear/additive predictors are to be plotted. The values must be from the set {1,2,...,r}. By default, all are plotted.

`control` Other control parameters. See `plotvgam.control`.

`...` Other arguments that can be fed into `plotvgam.control`. This includes line colors, line widths, line types, etc.

In the above, M is the number of linear/additive predictors, and r is the number of columns of the constraint matrix of interest.

Details

Many of `plotvgam()`'s options can be found in `plotvgam.control`, e.g., line types, line widths, colors.

Value

The original object, but with the `preplot` slot of the object assigned information regarding the plot.

Note

While `plot(fit)` will work if `class(fit)` is `"vgam"`, it is necessary to use `plotvgam(fit)` explicitly otherwise.

`plotvgam()` is quite buggy at the moment. `plotvgam()` works in a similar manner to S-PLUS's `plot.gam()`, however, there is no options for interactive construction of the plots yet.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Wild, C. J. (1996) Vector generalized additive models. *Journal of the Royal Statistical Society, Series B, Methodological*, **58**, 481–493.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

vgam, plotvgam.control, predict.vgam, vglm.

Examples

```
data(coalminers)
coalminers = transform(coalminers, Age = (age - 42) / 5)
fit = vgam(cbind(nBnW,nBW,BnW,BW) ~ s(Age), binom2.or(zero=NULL), coalminers)
## Not run:
par(mfrow=c(1,3))
plot(fit, se=TRUE, ylim=c(-3,2), las=1)

plot(fit, se=TRUE, which.cf=1:2, lcol="blue", scol="red", ylim=c(-3,2))
plot(fit, se=TRUE, which.cf=1:2, lcol="blue", scol="red", overlay=TRUE)
## End(Not run)
```

---

plotvgam.control *Control Function for plotvgam()*

Description

Provides default values for many arguments available for plotvgam().

Usage

```
plotvgam.control(which.cf = NULL,
                 xlim = NULL, ylim = NULL, llty = par()$lty,
                 slty = if (is.R()) "dashed" else 3, pcex = par()$cex,
                 pch = par()$pch, pcol = par()$col,
                 lcol = par()$col, rcol = par()$col,
                 scol = par()$col, llwd = par()$lwd, slwd = par()$lwd,
                 add.arg = FALSE, one.at.a.time = FALSE,
                 .include.dots = TRUE, ...)
```

Arguments

`which.cf`	Integer vector specifying which component functions are to be plotted (for each covariate). Must have values from the set {1,2,...,M}.
`xlim`	Range for the x-axis.
`ylim`	Range for the y-axis.
`llty`	Line type for the fitted functions (lines). Fed into `par(lty)`.
`slty`	Line type for the standard error bands. Fed into `par(lty)`.
`pcex`	Character expansion for the points (residuals). Fed into `par(cex)`.
`pch`	Character used for the points (residuals). Same as `par(pch)`.
`pcol`	Color of the points. Fed into `par(col)`.
`lcol`	Color of the fitted functions (lines). Fed into `par(col)`.
`rcol`	Color of the rug plot. Fed into `par(col)`.
`scol`	Color of the standard error bands. Fed into `par(col)`.
`llwd`	Line width of the fitted functions (lines). Fed into `par(lwd)`.
`slwd`	Line width of the standard error bands. Fed into `par(lwd)`.
`add.arg`	Logical. If `TRUE`, the plot will be added to an existing plot, otherwise a new plot will be made.
`one.at.a.time`	Logical. If `TRUE`, the plots are done one at a time, with the user having to hit the return key between the plots.
`.include.dots`	Not to be used by the user.
`...`	Other arguments that may be fed into `par()`.

In the above, M is the number of linear/additive predictors.

Details

The most obvious features of `plotvgam` can be controlled by the above arguments.

Value

A list with values matching the arguments.

Note

This function enables `plotvgam()` to work in a similar manner to S-PLUS's `plot.gam()`. However, there is no interactive options yet.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Wild, C. J. (1996) Vector generalized additive models. *Journal of the Royal Statistical Society, Series B, Methodological*, **58**, 481–493.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

plotvgam.

Examples

```
plotvgam.control(lcol=c("red", "blue"), scol="darkgreen", se=TRUE)
```

---

pneumo *Pneumoconiosis amongst a group of coalminers*

Description

The pneumo data frame has 8 rows and 4 columns. Exposure time is explanatory, and there are 3 ordinal response variables.

Usage

```
data(pneumo)
```

Format

This data frame contains the following columns:

exposure.time a numeric vector, in years

normal a numeric vector, counts

mild a numeric vector, counts

severe a numeric vector, counts

Details

These were collected from coalface workers. In the original data set, the two most severe categories were combined.

Source

Ashford, J.R., 1959. An approach to the analysis of data for semi-quantal responses in biological assay. *Biometrics*, **15**, 573–581.

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

Examples

```
data(pneumo)

# Fit the proportional odds model, p.179, in McCullagh and Nelder (1989)
pneumo = transform(pneumo, let=log(exposure.time))
vglm(cbind(normal,mild,severe) ~ let,
     cumulative(parallel=TRUE, reverse=TRUE), pneumo)
```

---

poissonff *Poisson Family Function*

Description

Family function for a generalized linear model fitted to Poisson responses. The dispersion parameters may be known or unknown.

Usage

```
poissonff(link = "loge", earg=list(), dispersion = 1,
          onedpar = FALSE, parallel = FALSE, zero = NULL)
```

Arguments

link	Link function applied to the mean or means. See Links for more choices.
earg	Extra argument optionally used by the link function. See Links for more information.
dispersion	Dispersion parameter. By default, maximum likelihood is used to estimate the model because it is known. However, the user can specify dispersion = 0 to have it estimated, or else specify a known positive value (or values if the response is a matrix—one value per column).
onedpar	One dispersion parameter? If the response is a matrix, then a separate dispersion parameter will be computed for each response (column), by default. Setting onedpar=TRUE will pool them so that there is only one dispersion parameter to be estimated.
parallel	A logical or formula. Used only if the response is a matrix.
zero	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,...,M}, where M is the number of columns of the matrix response.

Details

M defined above is the number of linear/additive predictors.

If the dispersion parameter is unknown, then the resulting estimate is not fully a maximum likelihood estimate.

A dispersion parameter that is less/greater than unity corresponds to under-/over-dispersion relative to the Poisson model. Over-dispersion is more common in practice.

When fitting a Quadratic RR-VGLM (see `cqo`), the response is a matrix of M, say, columns (e.g., one column per species). Then there will be M dispersion parameters (one per column of the response matrix) if `dispersion=0` and `onedpar=FALSE`.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `vgam`, `rrvglm`, `cqo`, and `cao`.

Warning

With a multivariate response, assigning a known dispersion parameter for *each* response is not handled well yet. Currently, only a single known dispersion parameter is handled well.

Note

This function will handle a matrix response automatically.

The call `poissonff(dispersion=0, ...)` is equivalent to `quasipoissonff(...)`. The latter was written so that R users of `quasipoisson()` would only need to add a "`ff`" to the end of the family function name.

Regardless of whether the dispersion parameter is to be estimated or not, its value can be seen from the output from the `summary()` of the object.

Author(s)

Thomas W. Yee

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

See Also

`Links`, `quasipoissonff`, `zipoisson`, `ordpoisson`, `loge`, `polf`, `rrvglm`, `cqo`, `cao`, `binomialff`, `quasibinomialff`, `poisson`, `poissonp`.

Examples

```
poissonff()

n = 100
x2 = rnorm(n)
x3 = rnorm(n)
x4 = rnorm(n)
lv1 = 0 + x3 - 2*x4
lambda1 = exp(3 - 0.5 * (lv1-0)^2)
lambda2 = exp(2 - 0.5 * (lv1-1)^2)
lambda3 = exp(2 - 0.5 * ((lv1+4)/2)^2)
y1 = rpois(n, lambda1)
y2 = rpois(n, lambda2)
y3 = rpois(n, lambda3)
p1 = cqo(cbind(y1,y2,y3) ~ x2 + x3 + x4, poissonff, EqualTol=FALSE,
         ITol=FALSE)
summary(p1)  # # Three dispersion parameters are all unity
## Not run:
lvplot(p1, y=TRUE, lcol=2:4, pch=2:4, pcol=2:4, rug=FALSE)
## End(Not run)
```

---

poissonp *Poisson-points-on-a-plane/volume Distances Distribution*

Description

Estimating the density parameter of the distances from a fixed point to the u-th nearest point, in a plane or volume.

Usage

```
poissonp(ostatistic, dimension=2, link="loge", earg=list(),
         idensity=NULL, method.init=1)
```

Arguments

ostatistic	Order statistic. A single positive integer. For example, the value 5 means the response are the distances of the fifth nearest value to that point (usually over many planes or volumes).
dimension	The value 2 or 3; 2 meaning a plane and 3 meaning a volume.
link	Parameter link function applied to the (positive) density parameter, called λ below. See Links for more choices.
earg	List. Extra argument for the link. See earg in Links for general information.
idensity	Optional initial value for the parameter. A NULL value means a value is obtained internally. Use this argument if convergence failure occurs.
method.init	An integer with value 1 or 2 which specifies the initialization method for λ. If failure to converge occurs try another value and/or else specify a value for idensity.

Details

Suppose the number of points in any region of area A of the plane is a Poisson random variable with mean λA (i.e., λ is the *density* of the points). Given a fixed point P, define D_1, D_2,... to be the distance to the nearest point to P, second nearest to P, etc. This **VGAM** family function estimates λ since the probability density function for D_u is easily derived, $u = 1, 2, \ldots$. Here, u corresponds to the argument `ostatistic`.

Similarly, suppose the number of points in any volume V is a Poisson random variable with mean λV where, once again, λ is the *density* of the points. This **VGAM** family function estimates λ by specifying the argument `ostatistic` and using `dimension=3`.

The mean of D_u is returned as the fitted values. Newton-Raphson is the same as Fisher-scoring.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

Warning

Convergence may be slow if the initial values are far from the solution. This often corresponds to the situation when the response values are all close to zero, i.e., there is a high density of points.

Formulae such as the means have not been fully checked.

Author(s)

T. W. Yee

See Also

`poissonff`.

Examples

```
y = rgamma(n <- 10, shape=exp(-1))   # Not good data!
os = 2
fit = vglm(y ~ 1, poissonp(os, 2), tra=TRUE, cri="c")
fit = vglm(y ~ 1, poissonp(os, 3), tra=TRUE, cri="c") # Slow convergence?
fit = vglm(y ~ 1, poissonp(os, 3, idensi=1), tra=TRUE, cri="c")
fitted(fit)[1:4]
mean(y)
coef(fit, matrix=TRUE)
Coef(fit)
```

polf *Poisson-Ordinal Link Function*

Description

Computes the Poisson-ordinal transformation, including its inverse and the first two derivatives.

Usage

```
polf(theta, earg = stop("'earg' must be given"), inverse = FALSE,
     deriv = 0, short = TRUE, tag = FALSE)
```

Arguments

theta	Numeric or character. See below for further details.
earg	Extra argument for passing in additional information. This must be list with component cutpoint. The cutpoints should be non-negative integers. If polf() is used as the link function in cumulative then one should choose reverse=TRUE, parallel=TRUE, intercept.apply=TRUE.
inverse	Logical. If TRUE the inverse function is computed.
deriv	Order of the derivative. Integer with value 0, 1 or 2.
short	Used for labelling the blurb slot of a vglmff-class object.
tag	Used for labelling the linear/additive predictor in the initialize slot of a vglmff-class object. Contains a little more information if TRUE.

Details

The Poisson-ordinal link function (POLF) can be applied to a parameter lying in the unit interval. Its purpose is to link cumulative probabilities associated with an ordinal response coming from an underlying Poisson distribution. If the cutpoint is zero then a complementary log-log link is used.

The arguments short and tag are used only if theta is character.

See Links for general information about **VGAM** link functions.

Value

See Yee (2006) for details.

Warning

Prediction may not work on vglm or vgam etc. objects if this link function is used.

Note

Numerical values of `theta` too close to 0 or 1 or out of range result in large positive or negative values, or maybe 0 depending on the arguments. Although measures have been taken to handle cases where `theta` is too close to 1 or 0, numerical instabilities may still arise.

In terms of the threshold approach with cumulative probabilities for an ordinal response this link function corresponds to the Poisson distribution (see `poissonff`) that has been recorded as an ordinal response using known cutpoints.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2006) *Ordinal ordination with normalizing link functions for count data*, (submitted for publication).

See Also

`Links`, `ordpoisson`, `poissonff`, `nbolf`, `golf`, `cumulative`.

Examples

```
earg = list(cutpoint=2)
polf("p", earg=earg, short=FALSE)
polf("p", earg=earg, tag=TRUE)

p = seq(0.01, 0.99, by=0.01)
y = polf(p, earg=earg)
y. = polf(p, earg=earg, deriv=1)
max(abs(polf(y, earg=earg, inv=TRUE) - p)) # Should be 0

## Not run:
par(mfrow=c(2,1), las=1)
plot(p, y, type="l", col="blue", main="polf()")
abline(h=0, v=0.5, col="red", lty="dashed")

plot(p, y., type="l", col="blue",
     main="(Reciprocal of) first POLF derivative")
## End(Not run)

# Rutherford and Geiger data
ruge = c(57,203,383,525,532,408,273,139,45,27,10,4,0,1,1)
y = 0:14
yy = rep(y, times=ruge)
length(yy)  # 2608 1/8-minute intervals
cutpoint = 5
yy01 = ifelse(yy <= cutpoint, 0, 1)
earg = list(cutpoint=cutpoint)
fit = vglm(yy01 ~ 1, binomialff(link="polf", earg=earg))
coef(fit, matrix=TRUE)
```

```
exp(coef(fit))

# Another example
nn = 1000
x2 = sort(runif(nn))
x3 = runif(nn)
mymu = exp( 3 + 1 * x2 - 2 * x3)
y1 = rpois(nn, lambda=mymu)
cutpoints = c(-Inf, 10, 20, Inf)
cuty = Cut(y1, breaks=cutpoints)
## Not run:
plot(x2, x3, col=cuty, pch=as.character(cuty))
## End(Not run)
table(cuty) / sum(table(cuty))
fit = vglm(cuty ~ x2 + x3, fam = cumulative(link="polf",
           reverse=TRUE, parallel=TRUE, intercept.apply=TRUE,
           mv=TRUE, earg=list(cutpoint=cutpoints[2:3])),
           trace=TRUE)
fit@y[1:5,]
fitted(fit)[1:5,]
predict(fit)[1:5,]
coef(fit)
coef(fit, matrix=TRUE)
constraints(fit)
fit@misc$earg
```

`Polono` *The Poisson Lognormal Distribution*

Description

Density, and random generation for the Poisson lognormal distribution.

Usage

```
dpolono(x, meanlog=0, sdlog=1, ...)
rpolono(n, meanlog=0, sdlog=1)
```

Arguments

`x`	vector of quantiles.
`n`	number of observations. Must be a positive integer of length 1.
`meanlog, sdlog`	the mean and standard deviation of the normal distribution (on the log scale). They match the arguments in `Lognormal`.
`...`	Arguments passed into `integrate`.

Details

The Poisson lognormal distribution is similar to the negative binomial in that it can be motivated by a Poisson distribution whose mean parameter comes from a right skewed distribution (gamma for the negative binomial and lognormal for the Poisson lognormal distribution).

Value

`dpolono` gives the density, and `rpolono` generates random deviates.

Note

`dpolono` involves numerical integration that is performed using `integrate`. Consequently, computations may be very slow. Also, numerical problems may occur, and if so, then the use of `...` may be needed.

For the maximum likelihood estimation of the 2 parameters a **VGAM** family function called `polono`, say, has not been written yet.

Author(s)

T. W. Yee

See Also

`lognormal`, `poissonff`, `negbinomial`.

Examples

```
## Not run:
meanlog = 0.5; sdlog = 0.5
y = 0:19
proby = dpolono(y, m=meanlog, sd=sdlog)
plot(y, proby, type="h", col="blue", las=1, ylab="P[Y=y]", log="",
     main=paste("Poisson lognormal(meanlog=",meanlog,", sdlog=",sdlog,")",
                sep=""))
sum(proby)  # Should be 1

y = rpolono(n=1000, m=meanlog, sd=sdlog)
table(y)
hist(y, breaks=((-1):max(y))+0.5, prob=TRUE)
## End(Not run)
```

Posbinom *Positive-Binomial Distribution*

Description

Density, distribution function, quantile function and random generation for the positive-binomial distribution.

Usage

```
dposbinom(x, size, prob, log = FALSE)
pposbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE)
qposbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)
rposbinom(n, size, prob)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. Must be a single positive integer.
`size`	number of trials. It is the N symbol in the formula given in `posbinomial`.
`prob`	probability of success on each trial.
`log, log.p, lower.tail`	Arguments that are passed on to `pbinom` etc.

Details

The positive-binomial distribution is a binomial distribution but with the probability of a zero being zero. The other probabilities are scaled to add to unity. The mean therefore is

$$\mu/(1-(1-\mu)^N)$$

where μ is the argument `prob` above. As μ increases, the positive-binomial and binomial distributions become more similar. Unlike similar functions for the binomial distribution, a zero value of `prob` is not permitted here.

Value

`dposbinom` gives the density, `pposbinom` gives the distribution function, `qposbinom` gives the quantile function, and `rposbinom` generates random deviates.

Note

For `rposbinom`, the arguments of the function are fed into `rbinom` until n positive values are obtained. This may take a long time if `prob` has values close to 0.

The family function `posbinomial` estimates the parameters by maximum likelihood estimation.

Author(s)

T. W. Yee

See Also

`posbinomial`, `rbinom`.

Examples

```
prob = 0.2
size = 10
y = rposbinom(n=1000, size, prob)
table(y)
mean(y)  # Sample mean
prob / (1-(1-prob)^size) # Population mean

(i = dposbinom(0:size, size, prob))
cumsum(i) - pposbinom(0:size, size, prob)  # Should be 0s
table(rposbinom(100, size, prob))

table(qposbinom(runif(1000), size, prob))
round(dposbinom(1:10, size, prob) * 1000) # Should be similar

## Not run:
x = 0:size
plot(x, dposbinom(x, size, prob), type="h", ylab="Probability",
     main=paste("Positive-binomial(", size, ",", prob, ") (blue) vs",
     " Binomial(", size, ",", prob, ") (red and shifted slightly)", sep=""),
     lwd=2, col="blue", las=1)
lines(x+0.05, dbinom(x, size, prob), type="h", lwd=2, col="red")
## End(Not run)
```

`posbinomial` *Positive Binomial Distribution Family Function*

Description

Fits a positive binomial distribution.

Usage

```
posbinomial(link = "logit", earg=list())
```

Arguments

`link`	Link function for the usual probability parameter. See `Links` for more choices.
`earg`	List. Extra argument for the link. See `earg` in `Links` for general information.

Details

The positive binomial distribution is the ordinary binomial distribution but with the probability of zero being zero. Thus the other probabilities are scaled up (i.e., divided by 1-P(Y=0)).

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

Warning

Under- or over-flow may occur if the data is ill-conditioned.

Note

The input for this family function is the same as `binomialff`.

Yet to be done: a `quasi.posbinomial` which estimates a dispersion parameter.

Author(s)

Thomas W. Yee

References

Patil, G. P. (1962) Maximum likelihood estimation for generalised power series distributions and its application to a truncated binomial distribution. *Biometrika*, **49**, 227–237.

Documentation accompanying the **VGAM** package at `http://www.stat.auckland.ac.nz/~yee` contains further information and examples.

See Also

`binomialff`.

Examples

```
# Number of albinotic children in families with 5 children
# Data from Patil (1962)
y = c(rep(1,25), rep(2,23), rep(3,10), 4, 5)  # No zeros
n = rep(5, 60)
yprop = y / 5
# Fit the identical models in two ways; MLE of p is 0.3088
fit = vglm(yprop ~ 1, posbinomial, trace=TRUE, weights=n)
fit2 = vglm(cbind(y, n-y) ~ 1, posbinomial, trace=TRUE)
summary(fit)
summary(fit2)
Coef(fit2)   # = MLE of p
Coef(fit)    # = MLE of p
fitted(fit2)[1:2]
fitted(fit)[1:2]
```

---

posnegbinomial *Positive Negative Binomial Distribution Family Function*

Description

Maximum likelihood estimation of the two parameters of a positive negative binomial distribution.

Usage

```
posnegbinomial(lmunb = "loge", lk = "loge",
               emunb =list(), ek = list(), ik = NULL,
               zero = -2, cutoff = 0.995, method.init=1)
```

Arguments

lmunb	Link function applied to the munb parameter, which is the mean μ_{nb} of an ordinary negative binomial distribution. See Links for more choices.
lk	Parameter link function applied to the dispersion parameter, called k. See Links for more choices.
emunb, ek	List. Extra argument for the respective links. See earg in Links for general information.
ik	Optional initial value for k, an index parameter. The value 1/k is known as a dispersion parameter. If failure to converge occurs try different values (and/or use method.init). If necessary this vector is recycled to length equal to the number of responses. A value NULL means an initial value for each response is computed internally using a range of values.
zero	Integer valued vector, usually assigned -2 or 2 if used at all. Specifies which of the two linear/additive predictors are modelled as an intercept only. By default, the k parameter (after lk is applied) is modelled as a single unknown number that is estimated. It can be modelled as a function of the explanatory variables by setting zero=NULL. A negative value means that the value is recycled, so setting -2 means all k are intercept only.
cutoff	A numeric which is close to 1 but never exactly 1. Used to specify how many terms of the infinite series are actually used. The sum of the probabilites are added until they reach this value or more. It is like specifying p in an imaginary function qnegbin(p).
method.init	See negbinomial.

Details

The positive negative binomial distribution is an ordinary negative binomial distribution but with the probability of a zero response being zero. The other probabilities are scaled to sum to unity.

This family function is based on negbinomial and most details can be found there. To avoid confusion, the parameter munb here corresponds to the mean of an ordinary negative binomial distribution negbinomial. The mean of posnegbinomial is

$$\mu_{nb}/(1 - p(0))$$

where $p(0) = (k/(k + \mu_{nb}))^k$ is the probability an ordinary negative binomial distribution has a zero value.

The parameters munb and k are not independent in the positive negative binomial distribution, whereas they are in the ordinary negative binomial distribution.

This function handles *multivariate* responses, so that a matrix can be used as the response. The number of columns is the number of species, say, and setting zero=-2 means that *all* species have a k equalling a (different) intercept only.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Warning

The Poisson model corresponds to k equalling infinity. If the data is Poisson or close to Poisson, numerical problems may occur. Possibly a loglog link could be added in the future to try help handle this problem.

Note

This family function can handle a multivariate response.

Author(s)

Thomas W. Yee

References

Barry, S. C. and Welsh, A. H. (2002) Generalized additive modelling and zero inflated count data. *Ecological Modelling*, **157**, 179–188.

See Also

rposnegbin, pospoisson, negbinomial, zanegbinomial, rnbinom.

Examples

```
## Not run:
x = runif(nn <- 2000)
y1 = rnbinom(nn, mu=exp(0+2*x), size=exp(1)) # k is size in rnbinom()
y2 = rnbinom(nn, mu=exp(1+2*x), size=exp(3))
fit = vglm(cbind(y1,y2) ~ 1, posnegbinomial, subset=(y1>0)&(y2>1),
           trace=TRUE)
coef(fit, matrix=TRUE)
dim(fit@y)
fitted(fit)[1:5,]
predict(fit)[1:5,]

# Another artificial data example
munb = exp(2); k = exp(3); n = 1000
y = rposnegbin(n, munb=munb, k=k)
table(y)
fit = vglm(y ~ 1, posnegbinomial, trace=TRUE)
coef(fit, matrix=TRUE)
mean(y)     # Sample mean
munb / (1 - (k/(k+munb))^k) # Population mean
fitted(fit)[1:5,]
predict(fit)[1:5,]
## End(Not run)
```

Posnorm *The Positive-Normal Distribution*

Description

Density, distribution function, quantile function and random generation for the univariate positive-normal distribution.

Usage

```
dposnorm(x, mean=0, sd=1)
pposnorm(q, mean=0, sd=1)
qposnorm(p, mean=0, sd=1)
rposnorm(n, mean=0, sd=1)
```

Arguments

x, q	vector of quantiles.
p	vector of probabilities.
n	number of observations. Must be a positive integer of length 1.
mean, sd	see rnorm.

Details

See posnormal1, the **VGAM** family function for estimating the parameters, for the formula of the probability density function and other details.

Value

dposnorm gives the density, pposnorm gives the distribution function, qposnorm gives the quantile function, and rposnorm generates random deviates.

Note

rposnorm() may run very slowly if the mean is very negative.

Author(s)

T. W. Yee

See Also

posnormal1.

Examples

```
## Not run:
m =  0.8
x = seq(-1, 4, len=501)
plot(x, dposnorm(x, m=m), type="l", ylim=0:1, las=1,
     ylab=paste("posnorm(m=",m,", sd=1)"), col="blue",
     main="Blue is density, red is cumulative distribution function",
     sub="Purple lines are the 10,20,...,90 percentiles")
lines(x, pposnorm(x, m=m), col="red")
abline(h=0)
probs = seq(0.1, 0.9, by=0.1)
Q = qposnorm(probs, m=m)
lines(Q, dposnorm(Q, m=m), col="purple", lty=3, type="h")
lines(Q, pposnorm(Q, m=m), col="purple", lty=3, type="h")
abline(h=probs, col="purple", lty=3)
pposnorm(Q, m=m) - probs # Should be all 0
## End(Not run)
```

`posnormal1` *Positive Normal Distribution Family Function*

Description

Fits a positive (univariate) normal distribution.

Usage

```
posnormal1(lmean="identity", lsd="loge",
           emean=list(), esd=list(),
           imean=NULL, isd=NULL, zero=NULL)
```

Arguments

`lmean, lsd`	Link functions for the mean and standard deviation parameters of the usual univariate normal distribution. They are μ and σ respectively. See `Links` for more choices.
`emean, esd`	List. Extra argument for each of the links. See `earg` in `Links` for general information.
`imean, isd`	Optional initial values for μ and σ. A `NULL` means a value is computed internally.
`zero`	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2} corresponding respectively to μ, σ. If `zero=NULL` then all linear/additive predictors are modelled as a linear combination of the explanatory variables. For many data sets having `zero=2` is a good idea.

Details

The positive normal distribution is the ordinary normal distribution but with the probability of zero or less being zero. The rest of the probability density function is scaled up. Hence the probability density function can be written

$$f(y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}(y-\mu)^2/\sigma^2\right) / \left[1 - \Phi(-\mu/\sigma)\right]$$

where $\Phi()$ is the cumulative distribution function of a standard normal (pnorm). Equivalently, this is

$$f(y) = \frac{1}{\sigma} \frac{\phi((y-\mu)/\sigma)}{1 - \Phi(-\mu/\sigma)}.$$

where $\phi()$ is the probability density function of a standard normal distribution (dnorm).

The mean of Y is

$$E(Y) = \mu + \sigma \frac{\phi(-\mu/\sigma)}{1 - \Phi(-\mu/\sigma)}.$$

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

Under- or over-flow may occur if the data is ill-conditioned.

Note

The response variable for this family function is the same as normal1 except positive values are required. Reasonably good initial values are needed. Fisher scoring is implemented.

Author(s)

Thomas W. Yee

References

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

normal1.

Examples

```
m =  1.0; SD = exp(1.0)
y = rposnorm(n <- 1000, m=m, sd=SD)
## Not run: hist(y, prob=TRUE, main=paste("posnorm(m=",m,", sd=",round(SD,2),")"))
fit = vglm(y ~ 1, fam=posnormal1, trace=TRUE)
coef(fit, mat=TRUE)
(Cfit = Coef(fit))
mygrid = seq(min(y), max(y), len=200) # Add the fit to the histogram
## Not run: lines(mygrid, dposnorm(mygrid, Cfit[1], Cfit[2]), col="red")
```

Pospois *Positive-Poisson Distribution*

Description

Density, distribution function, quantile function and random generation for the positive-Poisson distribution.

Usage

```
dpospois(x, lambda)
ppospois(q, lambda)
qpospois(p, lambda)
rpospois(n, lambda)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. Must be a single positive integer.
`lambda`	vector of positive means (of an ordinary Poisson distribution). Short vectors are recycled.

Details

The positive-Poisson distribution is a Poisson distribution but with the probability of a zero being zero. The other probabilities are scaled to add to unity. The mean therefore is

$$\lambda/(1-\exp(-\lambda)).$$

As λ increases, the positive-Poisson and Poisson distributions become more similar. Unlike similar functions for the Poisson distribution, a zero value of `lambda` is not permitted here.

Value

`dpospois` gives the density, `ppospois` gives the distribution function, `qpospois` gives the quantile function, and `rpospois` generates random deviates.

Note

For `rpospois`, the arguments of the function are fed into `rpois` until n positive values are obtained. This may take a long time if `lambda` has values close to 0.

The family function `pospoisson` estimates λ by maximum likelihood estimation.

Author(s)

T. W. Yee

See Also

`pospoisson`, `zapoisson`, `rpois`.

Examples

```
lambda = 2
y = rpospois(n=1000, lambda)
table(y)
mean(y)  # Sample mean
lambda / (1-exp(-lambda)) # Population mean

(i = dpospois(0:7, lambda))
cumsum(i) - ppospois(0:7, lambda)  # Should be 0s
table(rpospois(100, lambda))

table(qpospois(runif(1000), lambda))
round(dpospois(1:10, lambda) * 1000) # Should be similar

## Not run:
x = 0:7
plot(x, dpospois(x, lambda), type="h", ylab="Probability",
     main=paste("Positive Poisson(", lambda, ") (blue) vs",
     " Poisson(", lambda, ") (red and shifted slightly)", sep=""),
     lwd=2, col="blue", las=1)
lines(x+0.05, dpois(x, lambda), type="h", lwd=2, col="red")
## End(Not run)
```

---

pospoisson *Positive Poisson Distribution Family Function*

Description

Fits a positive Poisson distribution.

Usage

```
pospoisson(link = "loge", earg=list())
```

Arguments

link	Link function for the usual mean (lambda) parameter of an ordinary Poisson distribution. See Links for more choices.
earg	List. Extra argument for the link. See earg in Links for general information.

Details

The positive Poisson distribution is the ordinary Poisson distribution but with the probability of zero being zero. Thus the other probabilities are scaled up (i.e., divided by $1 - P[Y = 0]$). The mean, $\lambda/(1 - \exp(-\lambda))$, can be obtained by the extractor function fitted applied to the object.

A related distribution is the zero-inflated Poisson, in which the probability $P[Y = 0]$ involves another parameter ϕ. See zipoisson.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Warning

Under- or over-flow may occur if the data is ill-conditioned.

Note

Yet to be done: a quasi.pospoisson which estimates a dispersion parameter.

This family function can handle a multivariate response.

Author(s)

Thomas W. Yee

References

Coleman, J. S. and James, J. (1961) The equilibrium size distribution of freely-forming groups. *Sociometry*, **24**, 36–45.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

Pospois, posnegbinomial, poissonff, zipoisson.

Examples

```
y = 1:6
w = c(1486, 694, 195, 37, 10, 1) # Data from Coleman and James (1961)
fit = vglm(y ~ 1, pospoisson, weights=w)
Coef(fit)
summary(fit)
fitted(fit)

# Artificial data
x = runif(n <- 1000)
lambda = exp(2 + 3*x)
y = rpospois(n, lambda)
table(y)
fit = vglm(y ~ x, pospoisson, trace=TRUE, crit="c")
coef(fit, matrix=TRUE)
```

---

powl *Power Link Function*

Description

Computes the power transformation, including its inverse and the first two derivatives.

Usage

```
powl(theta, earg = list(power=1), inverse = FALSE, deriv = 0,
      short = TRUE, tag = FALSE)
```

Arguments

theta	Numeric or character. See below for further details.
earg	List. Extra argument for passing in additional information. Here, the component name power denotes the power or exponent. This component name should not be abbreviated.
inverse	Logical. If TRUE the inverse function is computed.
deriv	Order of the derivative. Integer with value 0, 1 or 2.
short	Used for labelling the blurb slot of a vglmff-class object.
tag	Used for labelling the linear/additive predictor in the initialize slot of a vglmff-class object. Contains a little more information if TRUE.

Details

The power link function raises a parameter by a certain value of power. Care is needed because it is very easy to get numerical problems, e.g., if power=0.5 and theta is negative.

The arguments short and tag are used only if theta is character.

Value

For `powl` with `deriv = 0`, then `theta` raised to the power of `power`. And if `inverse = TRUE` then `theta` raised to the power of `1/power`.

For `deriv = 1`, then the function returns *d* `theta` / *d* `eta` as a function of `theta` if `inverse = FALSE`, else if `inverse = TRUE` then it returns the reciprocal.

Note

Numerical problems may occur for certain combinations of `theta` and `power`. Consequently this link function should be used with caution.

Author(s)

Thomas W. Yee

See Also

`Links`, `loge`.

Examples

```
powl("a", earg=list(power=2), short=FALSE, tag=TRUE)

x = 1:5
powl(x)
powl(x, earg=list(power=2))

earg=list(power=2)
max(abs(powl(powl(x, earg=earg), earg=earg, inverse=TRUE) - x)) # Should be 0

x = (-5):5
powl(x, earg=list(power=0.5))  # Has NAs

# 1/2 = 0.5
y = rbeta(n=1000, shape1=2^2, shape2=3^2)
fit = vglm(y ~ 1, betaff(link="powl", earg=list(power=0.5), i1=3, i2=7),
           trace=TRUE, cri="coef")
coef(fit, matrix=TRUE)
Coef(fit)  # Useful for intercept-only models
vcov(fit, untrans=TRUE)
```

`predict.vglm` *Predict Method for a VGLM fit*

Description

Predicted values based on a vector generalized linear model (VGLM) object.

Usage

```
predict.vglm(object, newdata = NULL,
             type = c("link", "response", "terms"),
             se.fit = FALSE, deriv = 0, dispersion = NULL,
             untransform=FALSE, extra = object@extra, ...)
```

Arguments

object	Object of class inheriting from "vlm".
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
type	the type of prediction required. The default is the first one, meaning on the scale of the linear predictors. The alternative "response" is on the scale of the response variable, and depending on the family function, this may or may not be the mean. The "terms" option returns a matrix giving the fitted values of each term in the model formula on the linear predictor scale. The value of this argument can be abbreviated.
se.fit	logical: return standard errors?
deriv	Non-negative integer. Currently this must be zero. Later, this may be implemented for general values.
dispersion	Dispersion parameter. This may be inputted at this stage, but the default is to use the dispersion parameter of the fitted model.
extra	A list containing extra information. This argument should be ignored.
untransform	Logical. Reverses any parameter link function. This argument only works if type="link", se.fit=FALSE, deriv=0.
...	Arguments passed into predict.vlm.

Details

Obtains predictions and optionally estimates standard errors of those predictions from a fitted vector generalized linear model (VGLM) object.

This code implements *smart prediction* (see smartpred).

Value

If se.fit = FALSE, a vector or matrix of predictions. If se.fit = TRUE, a list with components

fitted.values	Predictions
se.fit	Estimated standard errors
df	Degrees of freedom
sigma	The square root of the dispersion parameter

Warning

This function may change in the future.

Note

Setting `se.fit=TRUE` and `type="response"` will generate an error.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

See Also

`predict`, `vglm`, `predict.vlm`, `smartpred`.

Examples

```
# Illustrates smart prediction
data(pneumo)
pneumo = transform(pneumo, let=log(exposure.time))
fit = vglm(cbind(normal,mild, severe) ~ poly(c(scale(let)), 2),
           fam=cumulative(parallel=TRUE),
           data=pneumo, trace=TRUE, x=FALSE)
class(fit)

(q0 = predict(fit)[1:3,])
(q1 = predict(fit, newdata=pneumo)[1:3,])
(q2 = predict(fit, newdata=pneumo[1:3,]))
all.equal(q0, q1)  # Should be TRUE
all.equal(q1, q2)  # Should be TRUE

predict(fit)[1:3,]
predict(fit, untransform=TRUE)[1:3,]

p0 = predict(fit, type="res")[1:3,]
p1 = predict(fit, type="res", newdata=pneumo)[1:3,]
p2 = predict(fit, type="res", newdata=pneumo[1:3,])
p3 = fitted(fit)[1:3,]
all.equal(p0, p1)  # Should be TRUE
all.equal(p1, p2)  # Should be TRUE
all.equal(p2, p3)  # Should be TRUE

predict(fit, type="t", se=TRUE)
```

`prentice74` *Prentice (1974) Log-gamma Distribution*

Description

Estimation of a 3-parameter log-gamma distribution described by Prentice (1974).

Usage

```
prentice74(llocation="identity", lscale="loge", lshape="identity",
           elocation=list(), escale=list(), eshape=list(),
           ilocation=NULL, iscale=NULL, ishape=NULL, zero=NULL)
```

Arguments

`llocation`	Parameter link function applied to the location parameter a. See `Links` for more choices.
`lscale`	Parameter link function applied to the positive scale parameter b. See `Links` for more choices.
`lshape`	Parameter link function applied to the shape parameter q. See `Links` for more choices.
`elocation, escale, eshape`	List. Extra argument for each of the links. See `earg` in `Links` for general information.
`ilocation, iscale`	Initial value for a and b, respectively. The defaults mean an initial value is determined internally for each.
`ishape`	Initial value for q. If failure to converge occurs, try some other value. The default means an initial value is determined internally.
`zero`	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,3}. The default value means none are modelled as intercept-only terms.

Details

The probability density function is given by

$$f(y; a, b, q) = |q| \exp(w/q^2 - e^w)/(b\Gamma(1/q^2)),$$

for shape parameter $q \neq 0$, positive scale parameter $b > 0$, location parameter a, and all real y. Here, $w = (y - a)q/b + \psi(1/q^2)$ where ψ is the digamma function. The mean of Y is a (returned as the fitted values). This is a different parameterization compared to `lgamma3ff`.

Special cases: $q = 0$ is the normal distribution with standard deviation b, $q = -1$ is the extreme value distribution for maxima, $q = 1$ is the extreme value distribution for minima (Weibull). If $q > 0$ then the distribution is left skew, else $q < 0$ is right skew.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

The special case $q = 0$ is not handled, therefore estimates of q too close to zero may cause numerical problems.

Note

The notation used here differs from Prentice (1974): $\alpha = a$, $\sigma = b$. Fisher scoring is used.

Author(s)

T. W. Yee

References

Prentice, R. L. (1974) A log gamma model and its maximum likelihood estimation. *Biometrika*, **61**, 539–544.

See Also

lgamma3ff, lgamma.

Examples

```
x = runif(n <- 5000)
loc = -1 + 2*x
Scale = exp(1+x)
y = rlgamma(n, loc=loc, scale=Scale, k=1)
fit = vglm(y ~ x, prentice74(zero=3), trace=TRUE)
coef(fit, matrix=TRUE)  # Note the coefficients for location
```

---

probit *Probit Link Function*

Description

Computes the probit transformation, including its inverse and the first two derivatives.

Usage

```
probit(theta, earg = list(), inverse = FALSE, deriv = 0,
       short = TRUE, tag = FALSE)
```

Arguments

theta	Numeric or character. See below for further details.
earg	Optional list. Extra argument for passing in additional information. Values of theta which are less than or equal to 0 can be replaced by the bvalue component of the list earg before computing the link function value. Values of theta which are greater than or equal to 1 can be replaced by 1 minus the bvalue component of the list earg before computing the link function value. The component name bvalue stands for "boundary value". See Links for general information about earg.
inverse	Logical. If TRUE the inverse function is computed.
deriv	Order of the derivative. Integer with value 0, 1 or 2.
short	Used for labelling the blurb slot of a vglmff-class object.
tag	Used for labelling the linear/additive predictor in the initialize slot of a vglmff-class object. Contains a little more information if TRUE.

Details

The probit link function is commonly used for parameters that lie in the unit interval. Numerical values of theta close to 0 or 1 or out of range result in Inf, -Inf, NA or NaN. The arguments short and tag are used only if theta is character.

Value

For deriv = 0, the probit of theta, i.e., qnorm(theta) when inverse = FALSE, and if inverse = TRUE then pnorm(theta).

For deriv = 1, then the function returns d theta / d eta as a function of theta if inverse = FALSE, else if inverse = TRUE then it returns the reciprocal.

Note

Numerical instability may occur when theta is close to 1 or 0. One way of overcoming this is to use earg.

In terms of the threshold approach with cumulative probabilities for an ordinal response this link function corresponds to the univariate normal distribution (see normal1).

Author(s)

Thomas W. Yee

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

See Also

Links, logit, cloglog, cauchit.

Examples

```
p = seq(0.01, 0.99, by=0.01)
probit(p)
max(abs(probit(probit(p), inverse=TRUE) - p)) # Should be 0

p = c(seq(-0.02, 0.02, by=0.01), seq(0.97, 1.02, by=0.01))
probit(p)  # Has NAs
probit(p, earg=list(bvalue= .Machine$double.eps))  # Has no NAs

## Not run:
plot(p, logit(p), type="l", col="limegreen", ylab="transformation",
     lwd=2, las=1, main="Some probability link functions")
lines(p, probit(p), col="purple", lwd=2)
lines(p, cloglog(p), col="chocolate", lwd=2)
lines(p, cauchit(p), col="tan", lwd=2)
abline(v=0.5, h=0, lty="dashed")
legend(0.1, 4.0, c("logit", "probit", "cloglog", "cauchit"),
       col=c("limegreen","purple","chocolate","tan"), lwd=2)
## End(Not run)
```

---

put.smart *Adds a List to the End of the List ".smart.prediction"*

Description

Adds a list to the end of the list `.smart.prediction` in `smartpredenv` (R) or frame 1 (S-PLUS).

Usage

```
put.smart(smart)
```

Arguments

`smart`	a list containing parameters needed later for smart prediction.

Details

`put.smart` is used in `"write"` mode within a smart function. It saves parameters at the time of model fitting, which are later used for prediction. The function `put.smart` is the opposite of `get.smart`, and both deal with the same contents.

Value

Nothing is returned.

Side Effects

The variable `.smart.prediction.counter` in `smartpredenv` (R) or frame 1 (S-PLUS) is incremented beforehand, and `.smart.prediction[[.smart.prediction.counter]]` is assigned the list `smart`. If the list `.smart.prediction` in `smartpredenv` (R) or frame 1 (S-PLUS) is not long enough to hold `smart`, then it is made larger, and the variable `.max.smart` in `smartpredenv` (R) or frame 1 (S-PLUS) is adjusted accordingly.

See Also

`get.smart`.

Examples

```
"my1" <- function(x, minx=min(x)) { # Here is a smart function
    x <- x  # Needed for nested calls, e.g., bs(scale(x))
    if(smart.mode.is("read")) {
        smart  <- get.smart()
        minx <- smart$minx          # Overwrite its value
    } else
    if(smart.mode.is("write"))
        put.smart(list(minx=minx))
    sqrt(x-minx)
}
attr(my1, "smart") <- TRUE
```

---

qrrvglm.control *Control function for QRR-VGLMs (CQO)*

Description

Algorithmic constants and parameters for a constrained quadratic ordination (CQO), by fitting a *quadratic reduced-rank vector generalized linear model* (QRR-VGLM), are set using this function. It is the control function for `cqo`.

Usage

```
qrrvglm.control(Rank = 1,
                Bestof = if(length(Cinit)) 1 else 10,
                checkwz=TRUE,
                Cinit = NULL,
                Crow1positive = TRUE,
                epsilon = 1.0e-06,
                EqualTolerances = ITolerances,
                Etamat.colmax = 10,
                FastAlgorithm = TRUE,
                GradientFunction=TRUE,
                Hstep = 0.001,
                isdlv = rep(c(2, 1, rep(0.5, len=Rank)), len=Rank),
```

```
iKvector = 0.1,
iShape = 0.1,
ITolerances = TRUE,
maxitl = 40,
method.init = 1,
Maxit.optim = 250,
MUXfactor = rep(7, length=Rank),
Norrr = ~ 1,
optim.maxit = 20,
Parscale = if(ITolerances) 0.001 else 1.0,
SD.Cinit = 0.02,
SmallNo = 5.0e-13,
trace = TRUE,
Use.Init.Poisson.QO=TRUE,
wzepsilon = .Machine$double.eps^0.75, ...)
```

Arguments

In the following, R is the `Rank`, M is the number of linear predictors, and S is the number of responses (species). Thus $M = S$ for binomial and Poisson responses, and $M = 2S$ for the negative binomial and 2-parameter gamma distributions.

`Rank` The numerical rank R of the model, i.e., the number of ordination axes. Must be an element from the set {1,2,...,min(M,p_2)} where the vector of explanatory variables x is partitioned into (x_1,x_2), which is of dimension $p_1 + p_2$. The variables making up x_1 are given by the terms in the `Norrr` argument, and the rest of the terms comprise x_2.

`Bestof` Integer. The best of `Bestof` models fitted is returned. This argument helps guard against local solutions by (hopefully) finding the global solution from many fits. The argument has value 1 if an initial value for C is inputted using `Cinit`.

`checkwz` logical indicating whether the diagonal elements of the working weight matrices should be checked whether they are sufficiently positive, i.e., greater than `wzepsilon`. If not, any values less than `wzepsilon` are replaced with this value.

`Cinit` Optional initial C matrix, which must be a p_2 by R matrix. The default is to apply `.Init.Poisson.QO()` to obtain initial values.

`Crow1positive` Logical vector of length `Rank` (recycled if necessary): are the elements of the first row of C positive? For example, if `Rank` is 4, then specifying `Crow1positive=c(FALSE, TRUE)` will force $C[1,1]$ and $C[1,3]$ to be negative, and $C[1,2]$ and $C[1,4]$ to be positive. This argument allows for a reflection in the ordination axes because the coefficients of the latent variables are unique up to a sign.

`epsilon` Positive numeric. Used to test for convergence for GLMs fitted in FORTRAN. Larger values mean a loosening of the convergence criterion. If an error code of 3 is reported, try increasing this value.

`EqualTolerances`

Logical indicating whether each (quadratic) predictor will have equal tolerances. Setting `EqualTolerances=TRUE` can help avoid numerical problems, especially with binary data. Note that the estimated (common) tolerance matrix may or may not be positive-definite. If it is, then it can be scaled to the R by R identity matrix, i.e., made equivalent to `ITolerances=TRUE`. Setting `ITolerances=TRUE` will *force* a common R by R identity matrix as the tolerance matrix to the data even if it is not appropriate. In general, setting `ITolerances=TRUE` is preferred over `EqualTolerances=TRUE` because, if it works, it is much faster and uses less memory. See **Details** for more details.

`Etamat.colmax`

Positive integer, no smaller than `Rank`. Controls the amount of memory used by `.Init.Poisson.QO()`. It is the maximum number of columns allowed for the pseudo-response and its weights. In general, the larger the value, the better the initial value. Used only if `Use.Init.Poisson.QO=TRUE`.

`FastAlgorithm`

Logical. Whether a new fast algorithm is to be used. The fast algorithm results in a large speed increases compared to Yee (2004). Some details of the fast algorithm are found in Appendix A of Yee (2006). Setting `FastAlgorithm=FALSE` will give an error.

`GradientFunction`

Logical. Whether `optim`'s argument `gr` is used or not, i.e., to compute gradient values. Used only if `FastAlgorithm` is `TRUE`. The default value is usually faster on most problems.

`Hstep`

Positive value. Used as the step size in the finite difference approximation to the derivatives by `optim`.

`isdlv`

Initial standard deviations for the latent variables (site scores). Numeric, positive and of length R (recycled if necessary). This argument is used only if `ITolerances=TRUE`. Used by `.Init.Poisson.QO()` to obtain initial values for the constrained coefficients C adjusted to a reasonable value. It adjusts the spread of the site scores relative to a common species tolerance of 1 for each ordination axis. A value between 0.5 and 10 is recommended; a value such as 10 means that the range of the environmental space is very large relative to the niche width of the species. The successive values should decrease because the first ordination axis should have the most spread of site scores, followed by the second ordination axis, etc.

`iKvector, iShape`

Numeric, recycled to length S if necessary. Initial values used for estimating the positive k and λ parameters of the negative binomial and 2-parameter gamma distributions respectively. For further information see `negbinomial` and `gamma2`. These arguments override the `ik` and `ishape` arguments in `negbinomial` and `gamma2`.

`ITolerances`

Logical. If `TRUE` then the (common) tolerance matrix is the R by R identity matrix by definition. Note that having `ITolerances=TRUE` implies `EqualTolerances=TRUE`, but not vice versa. Internally, the quadratic terms will be treated as offsets (in GLM jargon) and so the models can potentially be fitted very efficiently. *However, it is a very good idea to center all numerical variables in the x_2 vector*. See

Details for more details. The success of `ITolerances=TRUE` often depends on suitable values for `isdlv` and/or `MUXfactor`.

`maxitl` Maximum number of times the optimizer is called or restarted. Most users should ignore this argument.

`method.init` Method of initialization. A positive integer 1 or 2 or 3 etc. depending on the **VGAM** family function. Currently it is used for `negbinomial` and `gamma2` only, and used within the FORTRAN.

`Maxit.optim` Positive integer. Number of iterations given to the function `optim` at each of the `optim.maxit` iterations.

`MUXfactor` Multiplication factor for detecting large offset values. Numeric, positive and of length R (recycled if necessary). This argument is used only if `ITolerances=TRUE`. Offsets are -0.5 multiplied by the sum of the squares of all R latent variable values. If the latent variable values are too large then this will result in numerical problems. By too large, it is meant that the standard deviation of the latent variable values are greater than `MUXfactor[r] * isdlv[r]` for `r=1:Rank` (this is why centering and scaling all the numerical predictor variables in x_2 is recommended). A value about 3 or 4 is recommended. If failure to converge occurs, try a slightly lower value.

`optim.maxit` Positive integer. Number of times `optim` is invoked. At iteration `i`, the `i`th value of `Maxit.optim` is fed into `optim`.

`Norrr` Formula giving terms that are *not* to be included in the reduced-rank regression (or formation of the latent variables), i.e., those belong to x_1. Those variables which do not make up the latent variable (reduced-rank regression) correspond to the B_1 matrix. The default is to omit the intercept term from the latent variables.

`Parscale` Numerical and positive-valued vector of length C (recycled if necessary). Passed into `optim(..., control=list(parscale=Parscale))`; the elements of C become C / `Parscale`. Setting `ITolerances=TRUE` results in line searches that are very large, therefore C has to be scaled accordingly to avoid large step sizes. See **Details** for more information. It's probably best to leave this argument alone.

`SD.Cinit` Standard deviation of the initial values for the elements of C. These are normally distributed with mean zero. This argument is used only if `Use.Init.Poisson.QO = FALSE` and C is not inputted using `Cinit`.

`trace` Logical indicating if output should be produced for each iteration. The default is `TRUE` because the calculations are numerically intensive, meaning it may take a long time, so that the user might think the computer has locked up if `trace=FALSE`.

`SmallNo` Positive numeric between `.Machine$double.eps` and `0.0001`. Used to avoid under- or over-flow in the IRLS algorithm. Used only if `FastAlgorithm` is `TRUE`.

`Use.Init.Poisson.QO`
Logical. If `TRUE` then the function `.Init.Poisson.QO()` is used to obtain initial values for the canonical coefficients C. If `FALSE` then random numbers are used instead.

`wzepsilon`	Small positive number used to test whether the diagonals of the working weight matrices are sufficiently positive.
`...`	Ignored at present.

Details

Recall that the central formula for CQO is

$$\eta = B_1^T x_1 + A\nu + \sum_{m=1}^{M} (\nu^T D_m \nu) e_m$$

where x_1 is a vector (usually just a 1 for an intercept), x_2 is a vector of environmental variables, $\nu = C^T x_2$ is a R-vector of latent variables, e_m is a vector of 0s but with a 1 in the mth position. QRR-VGLMs are an extension of RR-VGLMs and allow for maximum likelihood solutions to constrained quadratic ordination (CQO) models.

Having `ITolerances=TRUE` means all the tolerance matrices are the order-R identity matrix, i.e., it *forces* bell-shaped curves/surfaces on all species. This results in a more difficult optimization problem (especially for 2-parameter models such as the negative binomial and gamma) because of overflow errors and it appears there are more local solutions. To help avoid the overflow errors, scaling C by the factor `Parscale` can help enormously. Even better, scaling C by specifying `isdlv` is more understandable to humans. If failure to converge occurs, try adjusting `Parscale`, or better, setting `EqualTolerances=TRUE` (and hope that the estimated tolerance matrix is positive-definite). To fit an equal-tolerances model, it is firstly best to try setting `ITolerances=TRUE` and varying `isdlv` and/or `MUXfactor` if it fails to converge. If it still fails to converge after many attempts, try setting `EqualTolerances=TRUE`, however this will usually be a lot slower because it requires a lot more memory.

With a $R > 1$ model, the latent variables are always uncorrelated, i.e., the variance-covariance matrix of the site scores is a diagonal matrix.

If setting `EqualTolerances=TRUE` is used and the common estimated tolerance matrix is positive-definite then that model is effectively the same as the `ITolerances=TRUE` model (the two are transformations of each other). In general, `ITolerances=TRUE` is numerically more unstable and presents a more difficult problem to optimize; the arguments `isdlv` and/or `MUXfactor` often must be assigned some good value(s) (possibly found by trial and error) in order for convergence to occur. Setting `ITolerances=TRUE` *forces* a bell-shaped curve or surface onto all the species data, therefore this option should be used with deliberation. If unsuitable, the resulting fit may be very misleading. Usually it is a good idea for the user to set `EqualTolerances=FALSE` to see which species appear to have a bell-shaped curve or surface. Improvements to the fit can often be achieved using transformations, e.g., nitrogen concentration to log nitrogen concentration.

Fitting a CAO model (see `cao`) first is a good idea for pre-examining the data and checking whether it is appropriate to fit a CQO model.

Value

A list with components matching the input names.

Warning

The default value of `Bestof` is a bare minimum for many datasets, therefore it will be necessary to increase its value to increase the chances of obtaining the global solution.

Note

When ITolerances=TRUE it is a good idea to apply scale to all the numerical variables that make up the latent variable, i.e., those of x_2. This is to make them have mean 0, and hence avoid large offset values which cause numerical problems.

This function has many arguments that are common with rrvglm.control and vglm.control.

It is usually a good idea to try fitting a model with ITolerances=TRUE first, and if convergence is unsuccessful, then try EqualTolerances=TRUE and ITolerances=FALSE. Ordination diagrams with EqualTolerances=TRUE have a natural interpretation, but with EqualTolerances=FALSE they are more complicated and requires, e.g., contours to be overlaid on the ordination diagram (see lvplot.qrrvglm).

In the example below, an equal-tolerances CQO model is fitted to the hunting spiders data. Because ITolerances=TRUE, it is a good idea to center all the x_2 variables first. Upon fitting the model, the actual standard deviation of the site scores are computed. Ideally, the isdlv argument should have had this value for the best chances of getting good initial values. For comparison, the model is refitted with that value and it should run more faster and reliably.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2004) A new technique for maximum-likelihood canonical Gaussian ordination. *Ecological Monographs*, **74**, 685–701.

Yee, T. W. (2006) Constrained additive ordination. *Ecology*, **87**, 203–213.

See Also

cqo, rcqo, Coef.qrrvglm, Coef.qrrvglm-class, optim, binomialff, poissonff, negbinomial, gamma2, gaussianff.

Examples

```
# Poisson CQO with equal tolerances
data(hspider)
set.seed(111)  # This leads to the global solution
hspider[,1:6]=scale(hspider[,1:6]) # Good idea when ITolerances = TRUE
p1 = cqo(cbind(Alopacce, Alopcune, Alopfabr, Arctlute, Arctperi, Auloalbi,
               Pardlugu, Pardmont, Pardnigr, Pardpull, Trocterr, Zoraspin) ~
         WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + ReflLux,
         ITolerances = TRUE,
         fam = quasipoissonff, data = hspider)
sort(p1@misc$deviance.Bestof) # A history of all the iterations

(isdlv = sd(lv(p1))) # should be approx isdlv

# Refit the model with better initial values
set.seed(111)  # This leads to the global solution
p1 = cqo(cbind(Alopacce, Alopcune, Alopfabr, Arctlute, Arctperi, Auloalbi,
```

```
                Pardlugu, Pardmont, Pardnigr, Pardpull, Trocterr, Zoraspin) ~
          WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + ReflLux,
          ITolerances = TRUE, isdlv = isdlv,   # Note the use of isdlv here
          fam = quasipoissonff, data = hspider)
sort(p1@misc$deviance.Bestof) # A history of all the iterations

# Negative binomial CQO; smallest deviance is about 275.389
set.seed(111)  # This leads to the global solution
nb1 = cqo(cbind(Alopacce, Alopcune, Alopfabr, Arctlute, Arctperi, Auloalbi,
                Pardlugu, Pardmont, Pardnigr, Pardpull, Trocterr, Zoraspin) ~
          WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + ReflLux,
          ITol = FALSE, EqualTol = TRUE, # A good idea for negbinomial
          fam = negbinomial, data = hspider)
sort(nb1@misc$deviance.Bestof) # A history of all the iterations
summary(nb1)
## Not run:
lvplot(nb1, lcol=1:12, y=TRUE, pcol=1:12)
## End(Not run)
```

`qtplot.gumbel` *Quantile Plot for Gumbel Regression*

Description

Plots quantiles associated with a Gumbel model.

Usage

```
qtplot.gumbel(object, plot.it = TRUE,
    y.arg = TRUE, spline.fit = FALSE, label = TRUE,
    R = object@misc$R, percentiles = object@misc$percentiles,
    add.arg = FALSE, mpv = object@misc$mpv,
    xlab = NULL, ylab = "", main = "",
    pch = par()$pch, pcol.arg = par()$col,
    llty.arg = par()$lty, lcol.arg = par()$col, llwd.arg = par()$lwd,
    tcol.arg = par()$col, tadj = 1, ...)
```

Arguments

`object`	A **VGAM** extremes model of the Gumbel type, produced by modelling functions such as `vglm` and `vgam` with a family function either `"gumbel"` or `"egumbel"`.
`plot.it`	Logical. Plot it? If `FALSE` no plot will be done.
`y.arg`	Logical. Add the raw data on to the plot?
`spline.fit`	Logical. Use a spline fit through the fitted percentiles? This can be useful if there are large gaps between some values along the covariate.
`label`	Logical. Label the percentiles?

R	See gumbel.
percentiles	See gumbel.
add.arg	Logical. Add the plot to an existing plot?
mpv	See gumbel.
xlab	Caption for the x-axis. See par.
ylab	Caption for the y-axis. See par.
main	Title of the plot. See title.
pch	Plotting character. See par.
pcol.arg	Color of the points. See the col argument of par.
llty.arg	Line type. Line type. See the lty argument of par.
lcol.arg	Color of the lines. See the col argument of par.
llwd.arg	Line width. See the lwd argument of par.
tcol.arg	Color of the text (if label is TRUE). See the col argument of par.
tadj	Text justification. See the adj argument of par.
...	Arguments passed into the plot function when setting up the entire plot. Useful arguments here include sub and las.

Details

There should be a single covariate such as time. The quantiles specified by percentiles are plotted.

Value

The object with a list called qtplot in the post slot of object. (If plot.it=FALSE then just the list is returned.) The list contains components

fitted.values	The percentiles of the response, possibly including the MPV.
percentiles	The percentiles (small vector of values between 0 and 100.

Note

Unlike gumbel, one cannot have percentiles=NULL.

Author(s)

Thomas W. Yee

See Also

gumbel.

Examples

```
data(venice)
y = as.matrix(venice[,paste("r",1:10,sep="")])
fit1 = vgam(y ~ s(year, df=3), gumbel(R=365, mpv=TRUE),
            data=venice, trace=TRUE, na.action=na.pass)
fitted(fit1)[1:4,]

## Not run:
par(mfrow=c(1,1), bty="l", xpd=TRUE, las=1)
qtplot(fit1, mpv=TRUE, lcol=c(1,2,5), tcol=c(1,2,5), lwd=2,
       pcol="blue", tadj=0.4)

qtplot(fit1, perc=97, mpv=FALSE, lcol=3, tcol=3,
       lwd=2, tadj=0.4, add=TRUE) -> i
i@post$qtplot$fitted[1:4,]
## End(Not run)
```

---

qtplot.lmscreg *Quantile Plot for LMS Quantile Regression*

Description

Plots quantiles associated with a LMS quantile regression.

Usage

```
qtplot.lmscreg(object, newdata = NULL,
               percentiles = object@misc$percentiles,
               plot.it = TRUE, ...)
```

Arguments

`object`	A **VGAM** quantile regression model, i.e., an object produced by modelling functions such as `vglm` and `vgam` with a family function beginning with `"lms."`, e.g., `lms.yjn`.
`newdata`	Optional data frame for computing the quantiles. If missing, the original data is used.
`percentiles`	Numerical vector with values between 0 and 100 that specify the percentiles (quantiles). The default are the percentiles used when the model was fitted.
`plot.it`	Logical. Plot it? If `FALSE` no plot will be done.
`...`	Graphical parameter that are passed into `plotqtplot.lmscreg`.

Details

The 'primary' variable is defined as the main covariate upon which the regression or smoothing is performed. For example, in medical studies, it is often the age. In **VGAM**, it is possible to handle more than one covariate, however, the primary variable must be the first term after the intercept.

Value

A list with the following components.

`fitted.values` A vector of fitted percentile values.

`percentiles` The percentiles used.

Note

`plotqtplot.lmscreg` does the actual plotting.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2004) Quantile regression via vector generalized additive models. *Statistics in Medicine*, **23**, 2295–2315.

Documentation accompanying the **VGAM** package at `http://www.stat.auckland.ac.nz/~yee` contains further information and examples.

See Also

`plotqtplot.lmscreg`, `deplot.lmscreg`, `lms.bcn`, `lms.bcg`, `lms.yjn`.

Examples

```
## Not run:
data(bminz)
fit = vgam(BMI ~ s(age, df=c(4,2)), fam=lms.bcn(zero=1), data=bminz)
qtplot(fit)
qtplot(fit, perc=c(25,50,75,95), lcol="blue", tcol="blue", llwd=2)
## End(Not run)
```

---

quasibinomialff *Quasi-Binomial Family Function*

Description

Family function for fitting generalized linear models to binomial responses, where the dispersion parameters are unknown.

Usage

```
quasibinomialff(link = "logit", mv = FALSE, onedpar = !mv,
                parallel = FALSE, zero = NULL)
```

Arguments

link	Link function. See Links for more choices.
mv	Multivariate response? If TRUE, then the response is interpreted as M binary responses, where M is the number of columns of the response matrix. In this case, the response matrix should have zero/one values only. If FALSE and the response is a (2-column) matrix, then the number of successes is given in the first column and the second column is the number of failures.
onedpar	One dispersion parameter? If mv, then a separate dispersion parameter will be computed for each response (column), by default. Setting onedpar=TRUE will pool them so that there is only one dispersion parameter to be estimated.
parallel	A logical or formula. Used only if mv is TRUE. This argument allows for the parallelism assumption whereby the regression coefficients for a variable is constrained to be equal over the M linear/additive predictors.
zero	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,...,M}, where M is the number of columns of the matrix response.

Details

The final model is not fully estimated by maximum likelihood since the dispersion parameter is unknown (see pp.124–8 of McCullagh and Nelder (1989) for more details).

A dispersion parameter that is less/greater than unity corresponds to under-/over-dispersion relative to the binomial model. Over-dispersion is more common in practice.

Setting mv=TRUE is necessary when fitting a Quadratic RR-VGLM (see cqo) because the response will be a matrix of M columns (e.g., one column per species). Then there will be M dispersion parameters (one per column of the response).

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, vgam, rrvglm, cqo, and cao.

Note

If mv is FALSE (the default), then the response can be of one of three formats: a factor (first level taken as success), a vector of proportions of success, or a 2-column matrix (first column = successes) of counts. The argument weights in the modelling function can also be specified. In particular, for a general vector of proportions, you will need to specify weights because the number of trials is needed.

If mv is TRUE, then the matrix response can only be of one format: a matrix of 1's and 0's (1=success).

This function is only a front-end to the **VGAM** family function binomialff(); indeed, quasibinomialff(...) is equivalent to binomialff(..., dispersion=0). Here, the argument dispersion=0 signifies that the dispersion parameter is to be estimated.

Regardless of whether the dispersion parameter is to be estimated or not, its value can be seen from the output from the summary() of the object.

Author(s)

Thomas W. Yee

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

See Also

binomialff, rrvglm, cqo, cao, logit, probit, cloglog, cauchit, poissonff, quasipoissonff, quasibinomial.

Examples

```
quasibinomialff()
quasibinomialff(link="probit")

# Nonparametric logistic regression
data(hunua)
hunua = transform(hunua, a.5 = sqrt(altitude))    # Transformation of altitude
fit1 = vglm(agaaus ~ poly(a.5, 2), quasibinomialff, hunua)
fit2 = vgam(agaaus ~ s(a.5, df=2), quasibinomialff, hunua)
## Not run:
plot(fit2, se=TRUE, llwd=2, lcol="red", scol="red",
     xlab="sqrt(altitude)", ylim=c(-3,1),
     main="GAM and quadratic GLM fitted to species data")
plotvgam(fit1, se=TRUE, lcol="blue", scol="blue", add=TRUE, llwd=2)
## End(Not run)
fit1@misc$dispersion   # dispersion parameter
logLik(fit1)

# Here, the dispersion parameter defaults to 1
fit0 = vglm(agaaus ~ poly(a.5, 2), binomialff, hunua)
fit0@misc$dispersion   # dispersion parameter
```

---

quasipoissonff *Quasi-Poisson Family Function*

Description

Fits a generalized linear model to a Poisson response, where the dispersion parameter is unknown.

Usage

```
quasipoissonff(link = "loge", onedpar = FALSE,
               parallel = FALSE, zero = NULL)
```

Arguments

link	Link function. See Links for more choices.
onedpar	One dispersion parameter? If the response is a matrix, then a separate dispersion parameter will be computed for each response (column), by default. Setting onedpar=TRUE will pool them so that there is only one dispersion parameter to be estimated.
parallel	A logical or formula. Used only if the response is a matrix.
zero	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,...,M}, where M is the number of columns of the matrix response.

Details

M defined above is the number of linear/additive predictors.

If the dispersion parameter is unknown, then the resulting estimate is not fully a maximum likelihood estimate.

A dispersion parameter that is less/greater than unity corresponds to under-/over-dispersion relative to the Poisson model. Over-dispersion is more common in practice.

When fitting a Quadratic RR-VGLM, the response is a matrix of M, say, columns (e.g., one column per species). Then there will be M dispersion parameters (one per column of the response matrix).

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, vgam, rrvglm, cqo, and cao.

Note

This function will handle a matrix response automatically.

The call poissonff(dispersion=0, ...) is equivalent to quasipoissonff(...). The latter was written so that R users of quasipoisson() would only need to add a "ff" to the end of the family function name.

Regardless of whether the dispersion parameter is to be estimated or not, its value can be seen from the output from the summary() of the object.

Author(s)

Thomas W. Yee

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

See Also

poissonff, loge, rrvglm, cqo, cao, binomialff, quasibinomialff, quasipoisson.

Examples

```
quasipoissonff()

## Not run:
n = 200; p = 5; S = 5
mydata = rcqo(n, p, S, fam="poisson", EqualTol=FALSE)
myform = attr(mydata, "formula")
p1 = cqo(myform, fam=quasipoissonff, EqualTol=FALSE, data=mydata)
sort(p1@misc$deviance.Bestof) # A history of all the iterations
lvplot(p1, y=TRUE, lcol=1:S, pch=1:S, pcol=1:S)
summary(p1)  # The dispersion parameters are estimated
## End(Not run)
```

---

rayleigh *Rayleigh Distribution Family Function*

Description

Estimating the parameter of the Rayleigh distribution by maximum likelihood estimation. Right-censoring is allowed.

Usage

```
rayleigh(link = "loge", earg=list(), nrfs=1/3+0.01)
crayleigh(link ="loge", earg=list(), expected=FALSE)
```

Arguments

link	Parameter link function applied to the parameter a. See Links for more choices. A log link is the default because a is positive.
earg	List. Extra argument for the link. See earg in Links for general information.
nrfs	Numeric, of length one, with value in $[0, 1]$. Weighting factor between Newton-Raphson and Fisher scoring. The value 0 means pure Newton-Raphson, while 1 means pure Fisher scoring. The default value uses a mixture of the two algorithms, and retaining positive-definite working weights.
expected	Logical. For censored data only, FALSE means the Newton-Raphson algorithm, and TRUE means Fisher scoring.

Details

The Rayleigh distribution, which is used in physics, has a probability density function that can be written

$$f(y) = y \exp(-0.5(y/a)^2)/a^2$$

for $y > 0$ and $a > 0$. The mean of Y is $a\sqrt{\pi/2}$ and its variance is $a^2(4 - \pi)/2$.

The **VGAM** family function crayleigh handles right-censored data (the true value is greater than the observed value). To indicate which type of censoring, input extra = list(rightcensored

= vec2) where vec2 is a logical vector the same length as the response. If the component of this list is missing then the logical values are taken to be FALSE. The fitted object has this component stored in the extra slot.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Warning

The theory behind the argument expected is not fully complete.

Note

A related distribution is the Maxwell distribution.

Author(s)

T. W. Yee

References

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

See Also

Rayleigh, maxwell.

Examples

```
n = 1000; a = exp(2)
ystar = rrayleigh(n, a=a)
fit = vglm(ystar ~ 1, rayleigh, trace=TRUE, crit="c")
fitted(fit)[1:5]
mean(ystar)
coef(fit, matrix=TRUE)
Coef(fit)

# Censored data
U = runif(n, 5, 15)
y = pmin(U, ystar)
## Not run:
par(mfrow=c(1,2)); hist(ystar); hist(y)
## End(Not run)
extra = list(rightcensored = ystar > U)
fit = vglm(y ~ 1, crayleigh, trace=TRUE, extra=extra)
table(fit@extra$rightcen)
coef(fit, matrix=TRUE)
fitted(fit)[1:4,]
```

rcqo *Constrained Quadratic Ordination*

Description

Random generation for constrained quadratic ordination (CQO).

Usage

```
rcqo(n, p, S, Rank = 1,
     family = c("poisson", "negbinomial", "binomial-poisson",
                "Binomial-negbinomial", "ordinal-poisson",
                "Ordinal-negbinomial", "gamma2"),
     EqualMaxima = FALSE, EqualTolerances = TRUE, ESOptima = FALSE,
     loabundance = if (EqualMaxima) hiabundance else 10,
     hiabundance = 100, sdlv = c(1.5/2^(0:3))[1:Rank],
     sdOptima = ifelse(ESOptima, 1.5/Rank, 1) * sdlv,
     sdTolerances = 0.25, Kvector = 1, Shape = 1,
     sqrt = FALSE, Log = FALSE, rhox = 0.5, breaks = 4,
     seed = NULL, Crow1positive=TRUE)
```

Arguments

n	Number of sites. It is denoted by n below.
p	Number of environmental variables, including an intercept term. It is denoted by p below. Must be no less than $1 + R$ in value.
S	Number of species. It is denoted by S below.
Rank	The rank or the number of latent variables or true dimension of the data on the reduced space. This must be either 1, 2, 3 or 4. It is denoted by R.
family	What type of species data is to be returned. The first choice is the default. If binomial then a 0 means absence and 1 means presence. If ordinal then the `breaks` argument is passed into the `breaks` argument of `cut`. Note that either the Poisson or negative binomial distributions are used to generate binomial and ordinal data, and that an upper-case choice is used for the negative binomial distribution (this makes it easier for the user). If `"gamma2"` then this is the 2-parameter gamma distribution.
EqualMaxima	Logical. Does each species have the same maxima? See arguments `loabundance` and `hiabundance`.
EqualTolerances	Logical. Does each species have the same tolerance? If `TRUE` then the common value is 1 along every latent variable, i.e., all species' tolerance matrices are the order-R identity matrix.
ESOptima	Logical. Do the species have equally spaced optima? If `TRUE` then the quantity $S^{1/R}$ must be an integer with value 2 or more. That is, there has to be an appropriate number of species in total. This is so that a grid of optimum values

is possible in R-dimensional latent variable space in order to place the species' optima. Also see the argument `sdTolerances`.

`loabundance, hiabundance`
Numeric. These are recycled to a vector of length S. The species have a maximum between `loabundance` and `hiabundance`. That is, at their optimal environment, the mean abundance of each species is between the two componentwise values. If `EqualMaxima` is `TRUE` then `loabundance` and `hiabundance` must have the same values. If `EqualMaxima` is `FALSE` then the logarithm of the maxima are uniformly distributed between `log(loabundance)` and `log(hiabundance)`.

`sdlv` Numeric, of length R (recycled if necessary). Site scores along each latent variable have these standard deviation values. This must be a decreasing sequence of values because the first ordination axis contains the greatest spread of the species' site scores, followed by the second axis, followed by the third axis, etc.

`sdOptima` Numeric, of length R (recycled if necessary). If `ESOptima=FALSE` then, for the rth latent variable axis, the optima of the species are generated from a normal distribution centered about 0. If `ESOptima=TRUE` then the S optima are equally spaced about 0 along every latent variable axis. Regardless of the value of `ESOptima`, the optima are then scaled to give standard deviation `sdOptima[r]`.

`sdTolerances` Logical. If `EqualTolerances=FALSE` then, for the rth latent variable, the species' tolerances are chosen from a normal distribution with mean 1 and standard deviation `sdTolerances[r]`. However, the first species `y1` has its tolerance matrix set equal to the order-R identity matrix. All tolerance matrices for all species are diagonal in this function. This argument is ignored if `EqualTolerances` is `TRUE`, otherwise it is recycled to length R if necessary.

`Kvector` A vector of positive k values (recycled to length S if necessary) for the negative binomial distribution (see `negbinomial` for details). Note that a natural default value does not exist, however the default value here is probably a realistic one, and that for large values of μ one has $Var(Y) = \mu^2/k$ approximately.

`Shape` A vector of positive λ values (recycled to length S if necessary) for the 2-parameter gamma distribution (see `gamma2` for details). Note that a natural default value does not exist, however the default value here is probably a realistic one, and that $Var(Y) = \mu^2/\lambda$.

`sqrt` Logical. Take the square-root of the negative binomial counts? Assigning `sqrt=TRUE` when `family="negbinomial"` means that the resulting species data can be considered very crudely to be approximately Poisson distributed. They will not integers in general but much easier (less numerical problems) to estimate using something like `cqo(..., family="poissonff")`.

`Log` Logical. Take the logarithm of the gamma random variates? Assigning `Log=TRUE` when `family="gamma2"` means that the resulting species data can be considered very crudely to be approximately Gaussian distributed about its (quadratic) mean. The result is that it is much easier (less numerical problems) to estimate using something like `cqo(..., family="gaussianff")`.

`rhox` Numeric, less than 1 in absolute value. The correlation between the environmental variables. The correlation matrix is a matrix of 1's along the diagonal and

rhox in the off-diagonals. Note that each environmental variable is normally distributed with mean 0. The standard deviation of each environmental variable is chosen so that the site scores have the determined standard deviation, as given by argument sdlv.

breaks If family is assigned an ordinal value then this argument is used to define the cutpoints. It is fed into the breaks argument of cut.

seed If given, it is passed into set.seed. This argument can be used to obtain reproducible results. If set, the value is saved as the "seed" attribute of the returned value. The default will not change the random generator state, and return .Random.seed as "seed" attribute.

Crow1positive
See qrrvglm.control for details.

Details

This function generates data coming from a constrained quadratic ordination (CQO) model. In particular, data coming from a *species packing model* can be generated with this function. The species packing model states that species have equal tolerances, equal maxima, and optima which are uniformly distributed over the latent variable space. This can be achieved by assigning the arguments ESOptima = TRUE, EqualMaxima = TRUE, EqualTolerances = TRUE.

At present, the Poisson and negative binomial abundances are generated first using loabundance and hiabundance, and if family is binomial or ordinal then it is converted into these forms.

In CQO theory the $n \times p$ matrix X is partitioned into two parts X_1 and X_2. The matrix X_2 contains the 'real' environmental variables whereas the variables in X_1 are just for adjustment purposes; they contain the intercept terms and other variables that one wants to adjust for when (primarily) looking at the variables in X_2. This function has X_1 only being a matrix of ones, i.e., containing an intercept only.

Value

A $n \times (p - 1 + S)$ data frame with components and attributes. In the following the attributes are labelled with double quotes.

x2, x3, x4, ..., xp
The environmental variables. This makes up the $n \times (p - 1)$ X_2 matrix. Note that x1 is not present; it is effectively a vector of ones since it corresponds to an intercept term when cqo is applied to the data.

y1, y2, x3, ..., yS
The species data. This makes up the $n \times S$ matrix Y. This will be of the form described by the argument family.

"ccoefficients"
The $(p - 1) \times R$ matrix of constrained coefficients (or canonical coefficients). These are also known as weights or loadings.

"formula" The formula involving the species and environmental variable names. This can be used directly in the formula argument of cqo.

"logmaxima" The S-vector of species' maxima, on a log scale. These are uniformly distributed between log(loabundance) and log(hiabundance).

"lv"	The $n \times R$ matrix of site scores. Each successive column (latent variable) has sample standard deviation equal to successive values of sdlv.
"optima"	The $S \times R$ matrix of species' optima.
"tolerances"	The $S \times R$ matrix of species' tolerances. These are the square root of the diagonal elements of the tolerance matrices (recall that all tolerance matrices are restricted to being diagonal in this function).

Other attributes are "break", "family", "Rank", "loabundance", "hiabundance", "EqualTolerances", "EqualMaxima", "seed" as used.

Note

This function is under development and is not finished yet. There may be a few bugs.

Yet to do: add an argument that allows absences to be equal to the first level if ordinal data is requested.

Author(s)

T. W. Yee

References

Yee, T. W. (2004) A new technique for maximum-likelihood canonical Gaussian ordination. *Ecological Monographs*, **74**, 685–701.

Yee, T. W. (2006) Constrained additive ordination. *Ecology*, **87**, 203–213.

ter Braak, C. J. F. and Prentice, I. C. (1988) A theory of gradient analysis. *Advances in Ecological Research*, **18**, 271–317.

See Also

cqo, qrrvglm.control, cut, binomialff, poissonff, negbinomial, gamma2, gaussianff.

Examples

```
# Example 1: Species packing model:
n = 100; p = 5; S = 5
mydata = rcqo(n, p, S, ESOpt=TRUE, EqualMax=TRUE)
names(mydata)
myform = attr(mydata, "formula")
fit = cqo(myform, fam=poissonff, ITol=TRUE, data=mydata,
          Bestof=3)  # Fit a CQO model to the data
## Not run:
matplot(attr(mydata, "lv"), mydata[,-(1:(p-1))], col=1:S)
persp(fit, col=1:S, add=TRUE)
lvplot(fit, lcol=1:S, y=TRUE, pcol=1:S)  # The same plot as above
## End(Not run)

# Compare the fitted model with the 'truth'
ccoef(fit)  # The fitted model
```

```
attr(mydata, "ccoefficients") # The 'truth'

c(sd(attr(mydata, "lv")), sd(lv(fit))) # Both values should be approx equal

# Example 2: negative binomial data fitted using a Poisson model:
n = 200; p = 5; S = 5
mydata = rcqo(n, p, S, fam="negbin", sqrt=TRUE)
myform = attr(mydata, "formula")
fit = cqo(myform, fam=poissonff, ITol=TRUE, dat=mydata)
## Not run:
lvplot(fit, lcol=1:S, y=TRUE, pcol=1:S)
## End(Not run)
# Compare the fitted model with the 'truth'
ccoef(fit)  # The fitted model
attr(mydata, "ccoefficients") # The 'truth'

# Example 3: gamma2 data fitted using a Gaussian model:
n = 200; p = 5; S = 3
mydata = rcqo(n, p, S, fam="gamma2", Log=TRUE)
fit = cqo(attr(mydata, "formula"), fam=gaussianff, ITol=TRUE, dat=mydata)
## Not run:
matplot(attr(mydata, "lv"), exp(mydata[,-(1:(p-1))]), col=1:S) # 'raw' data
lvplot(fit, lcol=1:S, y=TRUE, pcol=1:S)  # Fitted model to transformed data
## End(Not run)
# Compare the fitted model with the 'truth'
ccoef(fit)  # The fitted model
attr(mydata, "ccoefficients") # The 'truth'
```

`rdiric` *The Dirichlet distribution*

Description

Generates Dirichlet random variates.

Usage

```
rdiric(n, shape, dimension = NULL)
```

Arguments

`n`	number of observations.
`shape`	the shape parameters. These must be positive. If `dimension` is specifed, values are recycled if necessary to length `dimension`.
`dimension`	the dimension of the distribution. If `dimension` is not numeric then it is taken to be `length(shape)`.

Details

This function is based on a relationship between the gamma and Dirichlet distribution. Random gamma variates are generated, and then Dirichlet random variates are formed from these.

Value

A `n` by `dimension` matrix of Dirichlet random variates. Each element is positive, and each row will sum to unity.

Author(s)

Thomas W. Yee

References

Lange, K. (2002) *Mathematical and Statistical Methods for Genetic Analysis*, 2nd ed. New York: Springer-Verlag.

See Also

`dirichlet` is a **VGAM** family function for fitting a Dirichlet distribution to data.

Examples

```
y = rdiric(n=1000, shape=c(3, 1, 4))
fit = vglm(y ~ 1, dirichlet, trace = TRUE, crit="c")
Coef(fit)
coef(fit, matrix=TRUE)
```

---

recexp1 *Upper Record Values from a 1-parameter Exponential Distribution*

Description

Maximum likelihood estimation of the rate parameter of a 1-parameter exponential distribution when the observations are upper record values.

Usage

```
recexp1(lrate="loge", irate=NULL, method.init=1)
```

Arguments

`lrate`	Link function applied to the rate parameter. See `Links` for more choices.
`irate`	Numeric. Optional initial values for the rate. The default value `NULL` means they are computed internally, with the help of `method.init`.
`method.init`	Integer, either 1 or 2 or 3. Initial method, three algorithms are implemented. Choose the another value if convergence fails, or use `irate`.

Details

The response must be a vector or one-column matrix with strictly increasing values.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

By default, this family function has the intercept-only MLE as the initial value, therefore convergence may only take one iteration. Fisher scoring is used.

Author(s)

T. W. Yee

References

Arnold, B. C. and Balakrishnan, N. and Nagaraja, H. N. (1998) *Records*, New York: John Wiley & Sons.

See Also

`exponential`.

Examples

```
rawy = rexp(n <- 10000, rate=exp(1))

# Keep only the records
delete = c(FALSE, rep(TRUE, len=n-1))
for(i in 2:length(rawy))
    if(rawy[i] > max(rawy[1:(i-1)])) delete[i] = FALSE
(y = rawy[!delete])

length(y) / y[length(y)]   # MLE of rate

fit = vglm(y ~ 1, recexp1, trace=TRUE)
coef(fit, matrix=TRUE)
Coef(fit)
```

`reciprocal` *Reciprocal link function*

Description

Computes the reciprocal transformation, including its inverse and the first two derivatives.

Usage

```
reciprocal(theta, earg = list(), inverse.arg = FALSE, deriv = 0,
        short = TRUE, tag = FALSE)
nreciprocal(theta, earg = list(), inverse.arg = FALSE, deriv = 0,
        short = TRUE, tag = FALSE)
```

Arguments

`theta`	Numeric or character. See below for further details.
`earg`	Optional list. Extra argument for passing in additional information. Values of `theta` which are equal to 0 can be replaced by the `bvalue` component of the list `earg` before computing the link function value. The component name `bvalue` stands for "boundary value". See `Links` for general information about `earg`.
`inverse.arg`	Logical. If `TRUE` the inverse function is computed
`deriv`	Order of the derivative. Integer with value 0, 1 or 2.
`short`	Used for labelling the `blurb` slot of a `vglmff-class` object.
`tag`	Used for labelling the linear/additive predictor in the `initialize` slot of a `vglmff-class` object. Contains a little more information if `TRUE`.

Details

The `reciprocal` link function is a special case of the power link function. Numerical values of `theta` close to 0 result in `Inf`, `-Inf`, `NA` or `NaN`. The arguments `short` and `tag` are used only if `theta` is character.

The `nreciprocal` link function computes the negative reciprocal, i.e., $-1/\theta$.

Value

For `reciprocal`: for `deriv = 0`, the reciprocal of `theta`, i.e., `1/theta` when `inverse = FALSE`, and if `inverse = TRUE` then `1/theta`. For `deriv = 1`, then the function returns *d* `theta` / *d* `eta` as a function of `theta` if `inverse = FALSE`, else if `inverse = TRUE` then it returns the reciprocal.

Note

Numerical instability may occur when `theta` is close to 0.

Author(s)

Thomas W. Yee

References

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

See Also

identity, powl.

Examples

```
reciprocal(1:5)
reciprocal(1:5, inverse=TRUE, deriv=2)
nreciprocal(1:5)
nreciprocal(1:5, inverse=TRUE, deriv=2)

x = (-3):3
reciprocal(x)  # Has Inf
reciprocal(x, earg=list(bvalue= .Machine$double.eps))  # Has no Inf
```

---

recnormal1 *Upper Record Values from a Univariate Normal Distribution*

Description

Maximum likelihood estimation of the two parameters of a univariate normal distribution when the observations are upper record values.

Usage

```
recnormal1(lmean="identity", lsd="loge",
           imean=NULL, isd=NULL, method.init=1, zero=NULL)
```

Arguments

lmean, lsd	Link functions applied to the mean and sd parameters. See Links for more choices.
imean, isd	Numeric. Optional initial values for the mean and sd. The default value NULL means they are computed internally, with the help of method.init.
method.init	Integer, either 1 or 2 or 3. Initial method, three algorithms are implemented. Choose the another value if convergence fails, or use imean and/or isd.
zero	An integer vector, containing the value 1 or 2. If so, the mean or standard deviation respectively are modelled as an intercept only. Usually, setting zero=2 will be used, if used at all. The default value NULL means both linear/additive predictors are modelled as functions of the explanatory variables.

Details

The response must be a vector or one-column matrix with strictly increasing values.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

This family function tries to solve a difficult problem, and the larger the data set the better. Convergence failure can commonly occur, and convergence may be very slow, so set `maxit=200, trace=TRUE`, say. Inputting good initial values are advised.

This family function uses the BFGS quasi-Newton update formula for the working weight matrices. Consequently the estimated variance-covariance matrix may be inaccurate or simply wrong! The standard errors must be therefore treated with caution; these are computed in functions such as `vcov()` and `summary()`.

Author(s)

T. W. Yee

References

Arnold, B. C. and Balakrishnan, N. and Nagaraja, H. N. (1998) *Records*, New York: John Wiley & Sons.

See Also

`normal1`, `dcnormal1`.

Examples

```
n = 10000
mymean = 100
# First value is reference value or trivial record
rawy = c(mymean, rnorm(n, me=mymean, sd=16))

# Keep only observations that are records
delete = c(FALSE, rep(TRUE, len=n))
for(i in 2:length(rawy))
    if(rawy[i] > max(rawy[1:(i-1)])) delete[i] = FALSE
(y = rawy[!delete])

fit = vglm(y ~ 1, recnormal1, trace=TRUE, maxit=200)
coef(fit, matrix=TRUE)
Coef(fit)
summary(fit)
```

rhobit *Rhobit Link Function*

Description

Computes the rhobit link transformation, including its inverse and the first two derivatives.

Usage

```
rhobit(theta, earg = list(), inverse = FALSE, deriv = 0,
       short = TRUE, tag = FALSE)
```

Arguments

theta	Numeric or character. See below for further details.
earg	Optional list. Extra argument for passing in additional information. Values of theta which are less than or equal to -1 can be replaced by the bminvalue component of the list earg before computing the link function value. Values of theta which are greater than or equal to 1 can be replaced by the bmaxvalue component of the list earg before computing the link function value. See Links for general information about earg.
inverse	Logical. If TRUE the inverse function is computed.
deriv	Order of the derivative. Integer with value 0, 1 or 2.
short	Used for labelling the blurb slot of a vglmff-class object.
tag	Used for labelling the linear/additive predictor in the initialize slot of a vglmff-class object. Contains a little more information if TRUE.

Details

The rhobit link function is commonly used for parameters that lie between -1 and 1. Numerical values of theta close to -1 or 1 or out of range result in Inf, -Inf, NA or NaN. The arguments short and tag are used only if theta is character.

Value

For deriv = 0, the rhobit of theta, i.e., log((1 + theta)/(1 - theta)) when inverse = FALSE, and if inverse = TRUE then (exp(theta) - 1)/(exp(theta) + 1).

For deriv = 1, then the function returns d theta / d eta as a function of theta if inverse = FALSE, else if inverse = TRUE then it returns the reciprocal.

Note

Numerical instability may occur when theta is close to -1 or 1. One way of overcoming this is to use earg.

The correlation parameter of a standard bivariate normal distribution lies between -1 and 1, therefore this function can be used for modelling this parameter as a function of explanatory variables.

The link function rhobit is very similar to fisherz, e.g., just twice the value of fisherz.

Author(s)

Thomas W. Yee

References

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

Links, binom2.rho, fisherz.

Examples

```
theta = seq(-0.99, 0.99, by=0.01)
y = rhobit(theta)
## Not run:
plot(theta, y, type="l", las=1, ylab="", main="rhobit(theta)")
abline(v=0, h=0, lty=2)
## End(Not run)

x = c(seq(-1.02, -0.98, by=0.01), seq(0.97, 1.02, by=0.01))
rhobit(x)  # Has NAs
rhobit(x, earg=list(bminvalue= -1 + .Machine$double.eps,
                    bmaxvalue=  1 - .Machine$double.eps))  # Has no NAs
```

---

rig *Reciprocal Inverse Gaussian distribution*

Description

Estimation of the parameters of a reciprocal inverse Gaussian distribution.

Usage

```
rig(lmu = "identity", llambda = "loge",
    emu=list(), elambda=list(), imu = NULL, ilambda = 1)
```

Arguments

lmu, llambda	Link functions for mu and lambda. See Links for more choices.
imu, ilambda	Initial values for mu and lambda. A NULL means a value is computed internally.
emu, elambda	List. Extra argument for each of the links. See earg in Links for general information.

Details

See Jorgensen (1997) for details.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

This distribution is potentially useful for dispersion modelling.

Author(s)

T. W. Yee

References

Jorgensen, B. (1997) *The Theory of Dispersion Models*. London: Chapman & Hall

See Also

`simplex`.

Examples

```
y = rchisq(n=100, df=14)    # Not 'proper' data!!
fit = vglm(y ~ 1, rig, trace=TRUE)
fit = vglm(y ~ 1, rig, trace=TRUE, eps=1e-9, cri="c")
summary(fit)
```

---

rlplot.egev *Return Level Plot for GEV Fits*

Description

A return level plot is constructed for a GEV-type model.

Usage

```
rlplot.egev(object, plot.it = TRUE,
    probability = c((1:9)/100, (1:9)/10, 0.95, 0.99, 0.995, 0.999),
    add.arg = FALSE, xlab = "Return Period", ylab = "Return Level",
    main = "Return Level Plot",
    pch = par()$pch, pcol.arg = par()$col, pcex = par()$cex,
    llty.arg = par()$lty, lcol.arg = par()$col, llwd.arg = par()$lwd,
    slty.arg = par()$lty, scol.arg = par()$col, slwd.arg = par()$lwd,
    ylim = NULL, Log = TRUE, CI = TRUE, epsilon = 1e-05, ...)
```

Arguments

object	A **VGAM** extremes model of the GEV-type, produced by vglm with a family function either "gev" or "egev".
plot.it	Logical. Plot it? If FALSE no plot will be done.
probability	Numeric vector of probabilities used.
add.arg	Logical. Add the plot to an existing plot?
xlab	Caption for the x-axis. See par.
ylab	Caption for the y-axis. See par.
main	Title of the plot. See title.
pch	Plotting character. See par.
pcol.arg	Color of the points. See the col argument of par.
pcex	Character expansion of the points. See the cex argument of par.
llty.arg	Line type. Line type. See the lty argument of par.
lcol.arg	Color of the lines. See the col argument of par.
llwd.arg	Line width. See the lwd argument of par.
slty.arg, scol.arg, slwd.arg	Correponding arguments for the lines used for the confidence intervals. Used only if CI=TRUE.
ylim	Limits for the y-axis. Numeric of length 2.
Log	Logical. If TRUE then log="" otherwise log="x". This changes the labelling of the x-axis only.
CI	Logical. Add in a 95 percent confidence interval?
epsilon	Numeric, close to zero. Used for the finite-difference approximation to the first derivatives with respect to each parameter. If too small, numerical problems will occur.
...	Arguments passed into the plot function when setting up the entire plot. Useful arguments here include sub and las.

Details

A return level plot plots z_p versus $\log(y_p)$. It is linear if the shape parameter $\xi = 0$. If $\xi < 0$ then the plot is convex with asymptotic limit as p approaches zero at $\mu - \sigma/\xi$. And if $\xi > 0$ then the plot is concave and has no finite bound. Here, $G(z_p) = 1 - p$ where $0 < p < 1$ (p corresponds to the argument probability) and G is the cumulative distribution function of the GEV distribution. The quantity z_p is known as the *return level* associated with the *return period* $1/p$. For many applications, this means z_p is exceeded by the annual maximum in any particular year with probability p.

The points in the plot are the actual data.

Value

In the `post` slot of the object is a list called `rlplot` with list components

`yp`	`-log(probability)`, which is used on the x-axis.
`zp`	values which are used for the y-axis
`lower, upper`	lower and upper confidence limits for the 95 percent confidence intervals evaluated at the values of `probability` (if `CI=TRUE`).

Note

The confidence intervals are approximate, being based on finite-difference approximations to derivatives.

Author(s)

T. W. Yee

References

Coles, S. (2001) *An Introduction to Statistical Modeling of Extreme Values*. London: Springer-Verlag.

See Also

`egev`.

Examples

```
y = rgev(n <- 100, scale=exp(1), shape = -0.1)
fit = vglm(y ~ 1, egev, trace=TRUE)

# Identity link for all parameters:
fit2 = vglm(y ~ 1, egev(lsh=identity, lsc=identity, isc=10), trace=TRUE)
## Not run:
par(mfrow=c(1,2))
rlplot(fit) -> i1
rlplot(fit2, pcol="darkorange", lcol="blue", Log=FALSE,
       scol="darkgreen", slty="dashed") -> i2
range(i2@post$rlplot$upper - i1@post$rlplot$upper) # Should be near 0
range(i2@post$rlplot$lower - i1@post$rlplot$lower) # Should be near 0
## End(Not run)
```

`rposnegbin` *Positive-negative binomial distribution random variates*

Description

Generate random variates from a positive-negative binomial distribution.

Usage

```
rposnegbin(n, munb, k)
```

Arguments

`n`	number of random values to return.
`munb`	vector of positive means (of an ordinary negative binomial distribution).
`k`	vector of positive index parameters (of an ordinary negative binomial distribution). This is called the `size` argument in `rnbinom`. Short vectors are recycled. The parameter `1/k` is known as a dispersion parameter; as `k` approaches infinity, the negative binomial distribution approaches a Poisson distribution.

Details

The positive-negative binomial distribution is a negative binomial distribution but with the probability of a zero being zero. The other probabilities are scaled to add to unity. The mean therefore is

$$\mu/(1 - p(0))$$

where μ the mean of an ordinary negative binomial distribution. The arguments of the function are fed into `rnbinom` until n positive values are obtained.

Value

n random deviates are returned.

Note

The running time is slow when `munb` is very close to zero.

Author(s)

T. W. Yee

References

Welsh, A. H., Cunningham, R. B., Donnelly, C. F. and Lindenmayer, D. B. (1996) Modelling the abundances of rare species: statistical models for counts with extra zeros. *Ecological Modelling*, **88**, 297–308.

See Also

rnbinom, posnegbinomial, zanegbinomial.

Examples

```
munb = 2; k = 4; n = 1000
y = rposnegbin(n, munb=munb, k=k)
table(y)
mean(y)     # sample mean
munb / (1 - (k/(k+munb))^k) # population mean
```

---

rrar *Nested reduced-rank autoregressive models for multiple time series*

Description

Estimates the parameters of a nested reduced-rank autoregressive model for multiple time series.

Usage

```
rrar(Ranks = 1, coefstart = NULL)
```

Arguments

Ranks	Vector of integers: the ranks of the model. Each value must be at least one and no more than M, where M is the number of response variables in the time series. The length of Ranks is the *lag*, which is often denoted by the symbol L in the literature.
coefstart	Optional numerical vector of initial values for the coefficients. By default, the family function chooses these automatically.

Details

Full details are given in Ahn and Reinsel (1988). Convergence may be very slow, so setting maxits=50, say, may help. If convergence is not obtained, you might like to try inputting different initial values.

Setting trace=TRUE in vglm is useful for monitoring the progress at each iteration.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

This family function should be used within vglm and not with rrvglm because it does not fit into the RR-VGLM framework exactly. Instead, the reduced-rank model is formulated as a VGLM!

A methods function Coef.rrar, say, has yet to be written. It would return the quantities Ak1, C, D, omegahat, Phi, etc. as slots, and then print.Coef.rrar would also need to be written.

Author(s)

T. W. Yee

References

Ahn, S. and Reinsel, G. C. (1988) Nested reduced-rank autoregressive models for multiple time series. *Journal of the American Statistical Association*, **83**, 849–856.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

vglm, usagrain.

Examples

```
## Not run:
data(usagrain)
year = seq(1961+1/12, 1972+10/12, by=1/12)
par(mar=c(4,4,2,2)+0.1, mfrow=c(2,2))
for(i in 1:4) {
    plot(year, usagrain[,i], main=names(usagrain)[i], type="l", xlab="", ylab="")
    points(year, usagrain[,i], pch="*")
}
apply(usagrain, 2, mean)     # mu vector
cgrain = scale(usagrain, scale=FALSE) # Center the time series only
fit = vglm(cgrain ~ 1, rrar(Ranks=c(4,1)), trace=TRUE)
summary(fit)

print(fit@misc$Ak1, dig=2)
print(fit@misc$Cmatrices, dig=3)
print(fit@misc$Dmatrices, dig=3)
print(fit@misc$omegahat, dig=3)
print(fit@misc$Phimatrices, dig=2)

par(mar=c(4,4,2,2)+0.1, mfrow=c(4,1))
for(i in 1:4) {
    plot(year, fit@misc$Z[,i], main=paste("Z", i, sep=""),
         type="l", xlab="", ylab="")
    points(year, fit@misc$Z[,i], pch="*")
}
## End(Not run)
```

`rrvglm-class` *Class "rrvglm"*

Description

Reduced-rank vector generalized linear models.

Objects from the Class

Objects can be created by calls to `rrvglm`.

Slots

`extra`: Object of class `"list"`; the `extra` argument on entry to `vglm`. This contains any extra information that might be needed by the family function.

`family`: Object of class `"vglmff"`. The family function.

`iter`: Object of class `"numeric"`. The number of IRLS iterations used.

`predictors`: Object of class `"matrix"` with M columns which holds the M linear predictors.

`assign`: Object of class `"list"`, from class `"vlm"`. This named list gives information matching the columns and the (LM) model matrix terms.

`call`: Object of class `"call"`, from class `"vlm"`. The matched call.

`coefficients`: Object of class `"numeric"`, from class `"vlm"`. A named vector of coefficients.

`constraints`: Object of class `"list"`, from class `"vlm"`. A named list of constraint matrices used in the fitting.

`contrasts`: Object of class `"list"`, from class `"vlm"`. The contrasts used (if any).

`control`: Object of class `"list"`, from class `"vlm"`. A list of parameters for controlling the fitting process. See `vglm.control` for details.

`criterion`: Object of class `"list"`, from class `"vlm"`. List of convergence criterion evaluated at the final IRLS iteration.

`df.residual`: Object of class `"numeric"`, from class `"vlm"`. The residual degrees of freedom.

`df.total`: Object of class `"numeric"`, from class `"vlm"`. The total degrees of freedom.

`dispersion`: Object of class `"numeric"`, from class `"vlm"`. The scaling parameter.

`effects`: Object of class `"numeric"`, from class `"vlm"`. The effects.

`fitted.values`: Object of class `"matrix"`, from class `"vlm"`. The fitted values. This may be missing or consist entirely of `NA`s, e.g., the Cauchy model.

`misc`: Object of class `"list"`, from class `"vlm"`. A named list to hold miscellaneous parameters.

`model`: Object of class `"data.frame"`, from class `"vlm"`. The model frame.

`na.action`: Object of class `"list"`, from class `"vlm"`. A list holding information about missing values.

offset: Object of class "matrix", from class "vlm". If non-zero, a M-column matrix of offsets.

post: Object of class "list", from class "vlm" where post-analysis results may be put.

preplot: Object of class "list", from class "vlm" used by plotvgam; the plotting parameters may be put here.

prior.weights: Object of class "numeric", from class "vlm" holding the initially supplied weights.

qr: Object of class "list", from class "vlm". QR decomposition at the final iteration.

R: Object of class "matrix", from class "vlm". The **R** matrix in the QR decomposition used in the fitting.

rank: Object of class "integer", from class "vlm". Numerical rank of the fitted model.

residuals: Object of class "matrix", from class "vlm". The *working* residuals at the final IRLS iteration.

rss: Object of class "numeric", from class "vlm". Residual sum of squares at the final IRLS iteration with the adjusted dependent vectors and weight matrices.

smart.prediction: Object of class "list", from class "vlm". A list of data-dependent parameters (if any) that are used by smart prediction.

terms: Object of class "list", from class "vlm". The terms object used.

weights: Object of class "matrix", from class "vlm". The weight matrices at the final IRLS iteration. This is in matrix-band form.

x: Object of class "matrix", from class "vlm". The model matrix (LM, not VGLM).

xlevels: Object of class "list", from class "vlm". The levels of the factors, if any, used in fitting.

y: Object of class "matrix", from class "vlm". The response, in matrix form.

Extends

Class "vglm", directly. Class "vlm", by class "vglm".

Methods

biplot `signature(x = "rrvglm")`: biplot.

Coef `signature(object = "rrvglm")`: more detailed coefficients giving **A**, $\boldsymbol{B}_1$, **C**, etc.

biplot `signature(object = "rrvglm")`: biplot.

print `signature(x = "rrvglm")`: short summary of the object.

summary `signature(object = "rrvglm")`: a more detailed summary of the object.

Note

The slots of "rrvglm" objects are currently identical to "vglm" objects.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

Yee, T. W. and Wild, C. J. (1996) Vector generalized additive models. *Journal of the Royal Statistical Society, Series B, Methodological*, **58**, 481–493.

http://www.stat.auckland.ac.nz/~yee

See Also

rrvglm, lvplot.rrvglm, vglmff-class.

Examples

```
# Rank-1 stereotype model of Anderson (1984)
data(pneumo)
set.seed(111)
pneumo = transform(pneumo, let=log(exposure.time),
             x1=runif(nrow(pneumo))) # x1 is some unrelated covariate
fit = rrvglm(cbind(normal, mild, severe) ~ let + x1,
             multinomial, pneumo, Rank=1)
Coef(fit)
```

---

rrvglm *Fitting Reduced-Rank Vector Generalized Linear Models (RR-VGLMs)*

Description

A *reduced-rank vector generalized linear model* (RR-VGLM) is fitted. RR-VGLMs are VGLMs but some of the constraint matrices are estimated. In this documentation, M is the number of linear predictors.

Usage

```
rrvglm(formula, family, data = list(), weights = NULL, subset = NULL,
       na.action = na.fail, etastart = NULL, mustart = NULL,
       coefstart = NULL, control = rrvglm.control(...), offset = NULL,
       method = "rrvglm.fit", model = FALSE, x.arg = TRUE, y.arg = TRUE,
       contrasts = NULL, constraints = NULL, extra = NULL,
       qr.arg = FALSE, smart = TRUE, ...)
```

Arguments

formula	a symbolic description of the model to be fit. The RHS of the formula is applied to each linear predictor. Different variables in each linear predictor can be chosen by specifying constraint matrices.
family	a function of class "vglmff" describing what statistical model is to be fitted.

data	an optional data frame containing the variables in the model. By default the variables are taken from environment(formula), typically the environment from which rrvglm is called.
weights	an optional vector or matrix of (prior) weights to be used in the fitting process. If weights is a matrix, then it must be in *matrix-band* form, whereby the first M columns of the matrix are the diagonals, followed by the upper-diagonal band, followed by the band above that, etc. In this case, there can be up to $M(M+1)$ columns, with the last column corresponding to the (1,M) elements of the weight matrices.
subset	an optional logical vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The "factory-fresh" default is na.omit.
etastart	starting values for the linear predictors. It is a M-column matrix. If $M = 1$ then it may be a vector.
mustart	starting values for the fitted values. It can be a vector or a matrix. Some family functions do not make use of this argument.
coefstart	starting values for the coefficient vector.
control	a list of parameters for controlling the fitting process. See rrvglm.control for details.
offset	a vector or M-column matrix of offset values. These are *a priori* known and are added to the linear predictors during fitting.
method	the method to be used in fitting the model. The default (and presently only) method rrvglm.fit uses iteratively reweighted least squares (IRLS).
model	a logical value indicating whether the *model frame* should be assigned in the model slot.
x.arg, y.arg	logical values indicating whether the model matrix and response vector/matrix used in the fitting process should be assigned in the x and y slots. Note the model matrix is the LM model matrix; to get the VGLM model matrix type model.matrix(vglmfit) where vglmfit is a vglm object.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
constraints	an optional list of constraint matrices. The components of the list must be named with the term it corresponds to (and it must match in character format). Each constraint matrix must have M rows, and be of full-column rank. By default, constraint matrices are the M by M identity matrix unless arguments in the family function itself override these values. If constraints is used it must contain *all* the terms; an incomplete list is not accepted.
extra	an optional list with any extra information that might be needed by the family function.
qr.arg	logical value indicating whether the slot qr, which returns the QR decomposition of the VLM model matrix, is returned on the object.
smart	logical value indicating whether smart prediction (smartpred) will be used.
...	further arguments passed into rrvglm.control.

Details

The central formula is given by

$$\eta = B_1^T x_1 + A\nu$$

where x_1 is a vector (usually just a 1 for an intercept), x_2 is another vector of explanatory variables, and $\nu = C^T x_2$ is an R-vector of latent variables. Here, η is a vector of linear predictors, e.g., the mth element is $\eta_m = \log(E[Y_m])$ for the mth Poisson response. The matrices B_1, A and C are estimated from the data, i.e., contain the regression coefficients. For ecologists, the central formula represents a *constrained linear ordination* (CLO) since it is linear in the latent variables. It means that the response is a monotonically increasing or decreasing function of the latent variables.

The underlying algorithm of RR-VGLMs is iteratively reweighted least squares (IRLS) with an optimizing algorithm applied within each IRLS iteration (e.g., alternating algorithm).

In theory, any **VGAM** family function that works for `vglm` and `vgam` should work for `rrvglm` too.

`rrvglm.fit` is the function that actually does the work. It is `vglm.fit` with some extra code.

Value

An object of class `"rrvglm"`, which has the the same slots as a `"vglm"` object. The only difference is that the some of the constraint matrices are estimates rather than known. But **VGAM** stores the models the same internally. The slots of `"vglm"` objects are described in `vglm-class`.

Note

The smart prediction (`smartpred`) library is packed with the **VGAM** library.

The arguments of `rrvglm` are the same as those of `vglm` but with some extras in `rrvglm.control`.

In the example below, a rank-1 stereotype model of Anderson (1984) is fitted to some car data. The reduced-rank regression is performed, adjusting for two covariates. Setting a trivial constraint matrix for the latent variable variables in x_2 avoids a warning message when it is overwritten by a (common) estimated constraint matrix. It shows that German cars tend to be more expensive than American cars, given a car of fixed weight and width.

If `fit <- rrvglm(..., data=mydata)` then `summary(fit)` requires corner constraints and no missing values in `mydata`. Often the estimated variance-covariance matrix of the parameters is not positive-definite; if this occurs, try refitting the model with a different value for `Index.corner`.

For *constrained quadratic ordination* (CQO) see `cqo` for more details about QRR-VGLMs.

With multivariate binary responses, one must use `binomialff(mv=TRUE)` to indicate that the response (matrix) is multivariate. Otherwise, it is interpreted as a single binary response variable.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

Yee, T. W. (2004) A new technique for maximum-likelihood canonical Gaussian ordination. *Ecological Monographs*, **74**, 685–701.

Anderson, J. A. (1984) Regression and ordered categorical variables. *Journal of the Royal Statistical Society, Series B, Methodological*, **46**, 1–30.

See Also

`rrvglm.control`, `lvplot.rrvglm` (same as `biplot.rrvglm`), `rrvglm-class`, `grc`, `cqo`, `vglmff-class`, `vglm`, `vglm-class`, `smartpred`, `rrvglm.fit`. Methods functions include `Coef.rrvglm`, `summary.rrvglm`, etc.

Examples

```
data(car.all)
attach(car.all)
index = Country == "Germany" | Country == "USA" |
        Country == "Japan" | Country == "Korea"
detach(car.all)
scar = car.all[index, ]  # standardized car data
fcols = c(13,14,18:20,22:26,29:31,33,34,36)  # These are factors
scar[,-fcols] = scale(scar[,-fcols]) # Standardize all numerical vars
ones = matrix(1, 3, 1)
cms = list("(Intercept)"=diag(3), Width=ones, Weight=ones,
           Disp.=diag(3), Tank=diag(3), Price=diag(3),
           Frt.Leg.Room=diag(3))
set.seed(111)
fit = rrvglm(Country ~ Width + Weight + Disp. + Tank + Price + Frt.Leg.Room,
             multinomial, data =  scar, Rank = 2, trace = TRUE,
             constraints=cms, Norrr = ~ 1 + Width + Weight,
             Uncor=TRUE, Corner=FALSE, Bestof=2)
fit@misc$deviance  # A history of the fits
Coef(fit)
## Not run:
biplot(fit, chull=TRUE, scores=TRUE, clty=2, ccol="blue", scol="red",
       Ccol="darkgreen", Clwd=2, Ccex=2,
       main="1=Germany, 2=Japan, 3=Korea, 4=USA")
## End(Not run)
```

---

rrvglm.control *Control function for rrvglm*

Description

Algorithmic constants and parameters for running `rrvglm` are set using this function.

Usage

```
rrvglm.control(Rank = 1, Algorithm = c("alternating", "derivative"),
    Corner = TRUE, Uncor = FALSE, Wmat = NULL, Svd.arg = FALSE,
    Index.corner = if (length(Structural.zero))
    ((1:1000)[-Structural.zero])[1:Rank] else 1:Rank,
    Alpha = 0.5, Bestof = 1, Cinit = NULL,
    Etamat.colmax = 10,
    SD.Cinit = 0.02, Structural.zero = NULL,
    Norrr = ~1, trace = FALSE, Use.Init.Poisson.QO = FALSE,
    checkwz = TRUE, wzepsilon = .Machine$double.eps^0.75, ...)
```

Arguments

`Rank`	The numerical rank R of the model. Must be an element from the set {1,2,...,min(M,*p2*)}. Here, the vector of explanatory variables **x** is partitioned into (**x1**,**x2**), which is of dimension *p1*+*p2*. The variables making up **x1** are given by the terms in `Norrr` argument, and the rest of the terms comprise **x2**.
`Algorithm`	Character string indicating what algorithm is to be used. The default is the first one.
`Corner`	Logical indicating whether corner constraints are to be used. This is one method for ensuring a unique solution. If `TRUE`, `Index.corner` specifies the R rows of the constraint matrices that are use as the corner constraints, i.e., they hold an order-R identity matrix.
`Uncor`	Logical indicating whether uncorrelated latent variables are to be used. This is another normalization that forces the variance-covariance matrix of the latent variables to be `diag(Rank)`, i.e., unit variance and uncorrelated. This constraint does not lead to a unique solution because it can be rotated.
`Wmat`	Yet to be done.
`Svd.arg`	Logical indicating whether a singular value decomposition of the outer product is to computed. This is another normalization which ensures uniqueness. See the argument `Alpha` below.
`Index.corner`	Specifies the R rows of the constraint matrices that are used for the corner constraints, i.e., they hold an order-R identity matrix.
`Alpha`	The exponent in the singular value decomposition that is used in the first part: if the SVD is UDV^T then the first and second parts are UD^{α} and $D^{1-\alpha}V^T$ respectively. A value of 0.5 is 'symmetrical'. This argument is used only when `Svd.arg=TRUE`.
`Bestof`	Integer. The best of `Bestof` models fitted is returned. This argument helps guard against local solutions by (hopefully) finding the global solution from many fits. The argument works only when the function generates its own initial value for **C**, i.e., when **C** is *not* passed in as initial values.
`Cinit`	Initial **C** matrix which may speed up convergence. It must be of the correct dimension.
`Etamat.colmax`	Positive integer, no smaller than `Rank`. Controls the amount of memory used by `.Init.Poisson.QO()`. It is the maximum number of columns allowed for

the pseudo-response and its weights. In general, the larger the value, the better the initial value. Used only if `Use.Init.Poisson.QO=TRUE`.

`Structural.zero`
: Integer vector specifying which rows of the constraint matrices are to be all zeros.

`SD.Cinit`
: Standard deviation of the initial values for the elements of $\mathbf{C}$. These are normally distributed with mean zero. This argument is used only if `Use.Init.Poisson.QO = FALSE`.

`Norrr`
: Formula giving terms that are not to be included in the reduced-rank regression. These variables constitute the $\boldsymbol{B}_1$ matrix in the Yee and Hastie paper. Those variables which are subject to the reduced-rank regression correspond to the $\boldsymbol{B}_2$ matrix.

`trace`
: Logical indicating if output should be produced for each iteration.

`Use.Init.Poisson.QO`
: Logical indicating whether the `.Init.Poisson.QO()` should be used to obtain initial values for the $\mathbf{C}$. The function uses a new method that can work well if the data are Poisson counts coming from an equal-tolerances QRR-VGLM (CQO). This option is less realistic for RR-VGLMs compared to QRR-VGLMs.

`checkwz`
: logical indicating whether the diagonal elements of the working weight matrices should be checked whether they are sufficiently positive, i.e., greater than `wzepsilon`. If not, any values less than `wzepsilon` are replaced with this value.

`wzepsilon`
: Small positive number used to test whether the diagonals of the working weight matrices are sufficiently positive.

`...`
: Variables in ... are passed into `vglm.control`. If the derivative algorithm is used, then ... are also passed into `rrvglm.optim.control`.

In the above, R is the `Rank` and M is the number of linear predictors.

Details

VGAM supports three normalizations to ensure a unique solution. Of these, only corner constraints will work with `summary` of RR-VGLM objects.

Value

A list with components matching the input names. Some error checking is done, but not much.

Note

The arguments in this function begin with an upper case letter to help avoid interference with those of `vglm.control`.

In the example below a rank-1 stereotype model (Anderson, 1984) is fitted.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

See Also

rrvglm, rrvglm.optim.control, rrvglm-class, vglm, vglm.control, cqo.

Examples

```
data(pneumo)
set.seed(111)
pneumo = transform(pneumo, let=log(exposure.time),
            x1 = runif(nrow(pneumo)))  # x1 is some unrelated covariate
fit = rrvglm(cbind(normal, mild, severe) ~ let + x1,
             multinomial, pneumo, Rank=1, Index.corner=2)
constraints(fit)
vcov(fit)
summary(fit)
```

---

rrvglm.optim.control

Control function for rrvglm() calling optim()

Description

Algorithmic constants and parameters for running optim within rrvglm are set using this function.

Usage

```
rrvglm.optim.control(Fnscale = 1, Maxit = 100,
                     Switch.optimizer = 3, Abstol = -Inf,
                     Reltol = sqrt(.Machine$double.eps), ...)
```

Arguments

Fnscale	Passed into optim as fnscale.
Maxit	Passed into optim as maxit.
Switch.optimizer	Iteration number when the "Nelder-Mead" method of optim is switched to the quasi-Newton "BFGS" method. Assigning Switch.optimizer a negative number means always BFGS, while assigning Switch.optimizer a value greater than maxits means always use Nelder-Mead.
Abstol	Passed into optim as abstol.
Reltol	Passed into optim as reltol.
...	Ignored.

Details

See optim for more details.

Value

A list with components equal to the arguments.

Note

The transition between optimization methods may be unstable, so users may have to vary the value of Switch.optimizer.

Practical experience with Switch.optimizer shows that setting it to too large a value may lead to a local solution, whereas setting it to a low value will obtain the global solution. It appears that, if BFGS kicks in too late when the Nelder-Mead algorithm is starting to converge to a local solution, then switching to BFGS will not be sufficient to bypass convergence to that local solution.

Author(s)

Thomas W. Yee

See Also

rrvglm.control, optim.

---

ruge *Rutherford-Geiger Polonium Data*

Description

Decay counts of polonium recorded by Rutherford and Geiger (1910).

Usage

```
data(ruge)
```

Format

This data frame contains the following columns:

counts a numeric vector, counts or frequencies

number a numeric vector, the number of decays

Details

These are the radioactive decay counts of polonium recorded by Rutherford and Geiger (1910) representing the number of scintillations in 2608 1/8 minute intervals. For example, there were 57 frequencies of zero counts. The counts can be thought of as being approximately Poisson distributed.

Source

Rutherford, E. and Geiger, H. (1910) The Probability Variations in the Distribution of alpha Particles, *Philosophical Magazine*, **20**, 698–704.

Examples

```
data(ruge)
attach(ruge)
lambdahat = weighted.mean(number, w=counts)
(N = sum(counts))
cbind(number, counts, fitted=round(N * dpois(number, lam=lambdahat)))
detach(ruge)
```

---

s *Defining smooths in VGAM formulae*

Description

`s` is used in the definition of (vector) smooth terms within `vgam` formulae.

Usage

```
s(x, df = 4, spar = 0, ...)
```

Arguments

In the following, M is the number of additive predictors and r is the number of component functions to be estimated (so that r is an element from the set {1,2,...,M}). Also, if n is the number of *distinct* abscissae, then `s` will fail if $n < 7$.

`x`	covariate (abscissae) to be smoothed.
`df`	numerical vector of length r. Effective degrees of freedom: must lie between 1 (linear fit) and n (interpolation). Thus one could say that `df-1` is the *nonlinear degrees of freedom* of the smooth. Recycling of values will be used if `df` is not of length r.
`spar`	numerical vector of length r. Positive smoothing parameters (after scaling) . Larger values mean more smoothing so that the solution approaches a linear fit for that component function. A zero value means that `df` is used. Recycling of values will be used if `spar` is not of length r.
`...`	Ignored for now.

Details

`s`, which is symbolic and does not perform any smoothing itself, only handles a single covariate. It differs from the S-PLUS `s` and also the one from the `mgcv` library. They should not be mixed together.

S-PLUS's `s` allowed `spar` to be negative; **VGAM** does not allow this.

Value

A vector with attributes that are used by `vgam`.

Note

The `x` argument of `s()` must be a single variable and not a function of a variable. For example, `s(x)` is fine but `s(log(x))` will fail. In this case, let `logx <- log(x)`, say, and use `s(logx)`.

The vector cubic smoothing spline which `s()` represents is computationally demanding for large M. The cost is approximately $O(M^3)$.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Wild, C. J. (1996) Vector generalized additive models. *Journal of the Royal Statistical Society, Series B, Methodological*, **58**, 481–493.

See Also

`vgam`, `vsmooth.spline`.

Examples

```
# Nonparametric logistic regression
data(hunua)
fit = vgam(agaaus ~ s(altitude, df=2), binomialff, hunua)
## Not run:
plot(fit, se=TRUE)
## End(Not run)

# Bivariate logistic model with artificial data
n = 300
mydf = data.frame(x1=runif(n), x2=runif(n))
mydf = transform(mydf,
    y1 = ifelse(runif(n) < 1/(1+exp(-sin(2*x2))), 1, 0),
    y2 = ifelse(runif(n) < 1/(1+exp(-sin(2*x2))), 1, 0))
fit = vgam(cbind(y1,y2) ~ x1 + s(x2, 3), trace=TRUE,
           binom2.or(exchangeable = TRUE ~ s(x2,3)), data=mydf)
coef(fit, matrix=TRUE)
## Not run:
plot(fit, se=TRUE, which.term= 2, scol="blue")
## End(Not run)
```

seq2binomial *The Two-stage Sequential Binomial Distribution Family Function*

Description

Estimation of the probabilities of a two-stage binomial distribution.

Usage

```
seq2binomial(lprob1 = "logit", lprob2 = "logit", eprob1 = list(),
             eprob2 = list(), iprob1 = NULL, iprob2 = NULL, zero = NULL)
```

Arguments

`lprob1, lprob2`
: Parameter link functions applied to the two probabilities, called p and q below. See `Links` for more choices.

`eprob1, eprob2`
: Lists. Extra arguments for the links. See `earg` in `Links` for general information.

`iprob1, iprob2`
: Optional initial value for the first and second probabilities respectively. A `NULL` means a value is obtained in the `initialize` slot.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. If used, the value must be from the set {1,2} which correspond to the first and second probabilities respectively. A `NULL` value means none.

Details

This **VGAM** family function fits the model described by Crowder and Sweeting (1989) which is described as follows. Each of m spores has a probability p of germinating. Of the y_1 spores that germinate, each has a probability q of bending in a particular direction. Let y_2 be the number that bend in the specified direction. The probability model for this data is $P(y_1, y_2) =$

$$\binom{m}{y_1} p^{y_1} (1-p)^{m-y_1} \binom{y_1}{y_2} q^{y_2} (1-q)^{y_1-y_2}$$

for $0 < p < 1$, $0 < q < 1$, $y_1 = 1, \ldots, m$ and $y_2 = 1, \ldots, y_1$. Here, p is `prob1`, q is `prob2`.

Although the Authors refer to this as the *bivariate binomial* model, I have named it the *(two-stage) sequential binomial* model. Fisher scoring is used.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Note

The response must be a two-column matrix of sample proportions corresponding to y_1 and y_2. The m values should be inputted with the `weights` argument of `vglm` and `vgam`. The fitted value is a two-column matrix of estimated probabilities p and q.

Author(s)

Thomas W. Yee

References

Crowder, M. and Sweeting, T. (1989). Bayesian inference for a bivariate binomial distribution. *Biometrika*, **76**, 599–603.

See Also

`binomialff`.

Examples

```
mvector = round(rnorm(n <- 100, m=10, sd=2))
x = runif(n)
prob1 = logit(+2-x, inverse=TRUE)
prob2 = logit(-2+x, inverse=TRUE)
successes1 = rbinom(n=n, size=mvector, prob=prob1)
successes2 = rbinom(n=n, size=successes1, prob=prob2)
y1 = successes1 / mvector
y2 = successes2 / successes1
fit = vglm(cbind(y1,y2) ~ x, seq2binomial, trace=TRUE, weight=mvector)
coef(fit)
coef(fit, mat=TRUE)
fitted(fit)[1:5,]
```

`setup.smart` *Smart Prediction Setup*

Description

Sets up smart prediction in one of two modes: `"write"` and `"read"`.

Usage

```
setup.smart(mode.arg, smart.prediction=NULL, max.smart=30)
```

Arguments

mode.arg	mode.arg must be "write" or "read". If in "read" mode then smart.prediction must be assigned the data structure .smart.prediction that was created while fitting. This is stored in object@smart.prediction or object$smart.prediction where object is the name of the fitted object.
smart.prediction	If in "read" mode then smart.prediction must be assigned the list of data dependent parameters, which is stored on the fitted object. Otherwise, smart.prediction is ignored.
max.smart	max.smart is the initial length of the list .smart.prediction. It is not important because .smart.prediction is made larger if needed.

Details

This function is only required by programmers writing a modelling function such as lm and glm, or a prediction functions of such, e.g., predict.lm. The function setup.smart operates by mimicking the operations of a first-in first-out stack (better known as a *queue*).

Value

Nothing is returned.

Side Effects

In "write" mode .smart.prediction in smartpredenv (R) or frame 1 (S-PLUS) is assigned an empty list with max.smart components. In "read" mode .smart.prediction in smartpredenv (R) or frame 1 (S-PLUS) is assigned smart.prediction. In both cases, .smart.prediction.counter in smartpredenv (R) or frame 1 (S-PLUS) is assigned the value 0, and .smart.prediction.mode and .max.smart are written to smartpredenv (R) or frame 1 (S-PLUS) too.

See Also

lm, predict.lm.

Examples

```
## Not run:
# Put at the beginning of lm
setup.smart("write")
## End(Not run)

## Not run:
# Put at the beginning of predict.lm
setup.smart("read", smart.prediction=object$smart.prediction)
## End(Not run)
```

simplex *Simplex distribution*

Description

The two parameters of the univariate simplex distribution are estimated.

Usage

```
simplex(lmu = "logit", lsigma = "loge",
        emu=list(), esigma=list(),
        imu = NULL, isigma = NULL)
```

Arguments

lmu	Link function for mu. See Links for more choices. The parameter lies in the unit interval.
lsigma	Link function for sigma. See Links for more choices. The parameter is positive, therefore the log link is the default.
emu, esigma	List. Extra argument for each of the links. See earg in Links for general information.
imu, isigma	Optional initial values for mu and sigma. A NULL means a value is obtained internally.

Details

See Jorgensen (1997) for details.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

This distribution is potentially useful for dispersion modelling. This family function only works for intercept-only models, i.e., the formula should have ~ 1.

Author(s)

T. W. Yee

References

Jorgensen, B. (1997) *The Theory of Dispersion Models*. London: Chapman & Hall

See Also

rig.

Examples

```
x = runif(n <- 100)
y = rbeta(n, shape1=3+x, shape2=4-x)
(fit = vglm(y ~ 1, simplex, trace=TRUE, cri="coef"))
coef(fit, matrix=TRUE)
Coef(fit)
summary(fit)
```

---

sinmad *Singh-Maddala Distribution Family Function*

Description

Maximum likelihood estimation of the 3-parameter Singh-Maddala distribution.

Usage

```
sinmad(link.a = "loge", link.scale = "loge", link.q = "loge",
       earg.a=list(), earg.scale=list(), earg.q=list(),
       init.a = NULL, init.scale = NULL, init.q = 1, zero = NULL)
```

Arguments

`link.a, link.scale, link.q`
: Parameter link functions applied to the (positive) parameters `a`, `scale`, and `q`. See `Links` for more choices.

`earg.a, earg.scale, earg.q`
: List. Extra argument for each of the links. See `earg` in `Links` for general information.

`init.a, init.scale, init.q`
: Optional initial values for `a`, `scale`, and `q`.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. Here, the values must be from the set {1,2,3} which correspond to `a`, `scale`, `q`, respectively.

Details

The 3-parameter Singh-Maddala distribution is the 4-parameter generalized beta II distribution with shape parameter $p = 1$. It is known under various other names, such as the Burr XII (or just the Burr distribution), Pareto IV, beta-P, and generalized log-logistic distribution. More details can be found in Kleiber and Kotz (2003).

Some distributions which are special cases of the 3-parameter Singh-Maddala are the Lomax ($a = 1$), Fisk ($q = 1$), and paralogistic ($a = q$).

The Singh-Maddala distribution has density

$$f(y) = aqy^{a-1}/[b^a\{1+(y/b)^a\}^{1+q}]$$

for $a > 0$, $b > 0$, $q > 0$, $y > 0$. Here, b is the scale parameter `scale`, and the others are shape parameters. The cumulative distribution function is

$$F(y) = 1 - [1+(y/b)^a]^{-q}.$$

The mean is

$$E(Y) = b\,\Gamma(1+1/a)\,\Gamma(q-1/a)/\Gamma(q)$$

provided $-a < 1 < aq$.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

If the self-starting initial values fail, try experimenting with the initial value arguments, especially those whose default value is not `NULL`.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

See Also

`Sinmad`, `genbetaII`, `betaII`, `dagum`, `fisk`, `invlomax`, `lomax`, `paralogistic`, `invparalogistic`.

Examples

```
y = rsinmad(n=3000, 3, 5, 2)
fit = vglm(y ~ 1, sinmad, trace=TRUE)
fit = vglm(y ~ 1, sinmad, trace=TRUE, crit="c")
coef(fit, mat=TRUE)
Coef(fit)
summary(fit)
```

skewnormal1 *Univariate Skew-Normal Distribution Family Function*

Description

Maximum likelihood estimation of the shape parameter of a univariate skew-normal distribution.

Usage

```
skewnormal1(lshape = "identity", earg = list(), ishape = NULL)
```

Arguments

lshape	Link function applied to the shape parameter. See Links for more choices.
earg	List. Extra argument for the link. See earg in Links for general information.
ishape	Optional inital value for the shape parameter. The default is to choose one internally. See the note below.

Details

The univariate skew-normal distribution has a density function that can be written

$$f(y) = 2\,\phi(y)\,\Phi(\alpha y)$$

where α is the shape parameter. Here, ϕ is the standard normal density and Φ its cumulative distribution function. When $\alpha = 0$ the result is a standard normal distribution. When $\alpha = 1$ it models the distribution of the maximum of two independent standard normal variates. When the absolute value of the shape parameter increases the skewness of the distribution increases. The limit as the shape parameter tends to positive infinity results in the folded normal distribution or half-normal distribution. When the shape parameter changes its sign, the density is reflected about $y = 0$.

The mean of the distribution is $\mu = \alpha\sqrt{2/(\pi(1+\alpha^2))}$ and these are returned as the fitted values. The variance of the distribution is $1 - \mu^2$. The Newton-Raphson algorithm is used.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

It is a good idea to use several different initial values to ensure that the global solution is obtained.

This family function will be modified (hopefully soon) to handle a location and scale parameter too.

Author(s)

Thomas W. Yee

References

Azzalini, A. A. (1985) A class of distributions which include the normal. *Scandinavian Journal of Statistics*, **12**, 171–178.

Azzalini, A. and Capitanio, A. (1999) Statistical applications of the multivariate skew-normal distribution. *Journal of the Royal Statistical Society, Series B, Methodological*, **61**, 579–602.

See Also

snorm, normal1.

Examples

```
y = rsnorm(n <- 1000, shape=5)
fit = vglm(y ~ 1, skewnormal1, trace=TRUE)
coef(fit, matrix=TRUE)
fitted(fit)[1:4,]
mean(y)
## Not run:
hist(y, prob=TRUE)
x = seq(min(y), max(y), len=200)
lines(x, dsnorm(x, shape=Coef(fit)), col="blue")
## End(Not run)

x = runif(n)
y = rsnorm(n, shape=1 + 2*x)
fit = vglm(y ~ x, skewnormal1, trace=TRUE, crit="coef")
summary(fit)
```

---

smart.expression *S Expression for Smart Functions*

Description

`smart.expression` is an S expression for a smart function to call itself. It is best if you go through it line by line, but most users will not need to know anything about it. It requires the primary argument of the smart function to be called `"x"`.

The list component `match.call` must be assigned the value of `match.call()` in the smart function; this is so that the smart function can call itself later.

See Also

match.call.

Examples

```
"my2" <- function(x, minx=min(x)) { # Here is a smart function
    x <- x  # Needed for nested calls, e.g., bs(scale(x))
    if(smart.mode.is("read")) {
        return(eval(smart.expression))
    } else
    if(smart.mode.is("write"))
        put.smart(list(minx=minx, match.call=match.call()))
    (x-minx)^2
}
attr(my2, "smart") <- TRUE
```

---

smart.mode.is *Determine What Mode the Smart Prediction is In*

Description

Determine which of three modes the smart prediction is currently in.

Usage

```
smart.mode.is(mode.arg=NULL)
```

Arguments

mode.arg	a character string, either "read", "write" or "neutral".

Details

Smart functions such as bs and poly need to know what mode smart prediction is in. If it is in "write" mode then the parameters are saved to .smart.prediction using put.smart. If in "read" mode then the parameters are read in using get.smart. If in "neutral" mode then the smart function behaves like an ordinary function.

Value

If mode.arg is given, then either TRUE or FALSE is returned. If mode.arg is not given, then the mode ("neutral", "read" or "write") is returned. Usually, the mode is "neutral".

See Also

put.smart, bs, poly.

Examples

```
my1 <- function(x, minx = min(x)) { # Here is a smart function
    x <- x  # Needed for nested calls, e.g., bs(scale(x))
    if(smart.mode.is("read")) {
        smart <- get.smart()
        minx <- smart$minx      # Overwrite its value
    } else if(smart.mode.is("write"))
              put.smart(list(minx = minx))
    sqrt(x - minx)
}
attr(my1, "smart") <- TRUE

smart.mode.is()    # Returns "neutral"
smart.mode.is(smart.mode.is())  # Returns TRUE
```

---

smartpred *Smart Prediction*

Description

Data-dependent parameters in formula terms can cause problems in when predicting. The **smartpred** package for R and S-PLUS saves data-dependent parameters on the object so that the bug is fixed. The `lm` and `glm` functions have been fixed properly. Note that the **VGAM** package by T. W. Yee automatically comes with smart prediction.

Details

R version 1.6.0 introduced a partial fix for the prediction problem because it does not work all the time, e.g., for terms such as `I(poly(x, 3))`, `poly(c(scale(x)), 3)`, `bs(scale(x), 3)`, `scale(scale(x))`. See the examples below. Smart prediction, however, will always work.

The basic idea is that the functions in the formula are now smart, and the modelling functions make use of these smart functions. Smart prediction works in two ways: using `smart.expression`, or using a combination of `put.smart` and `get.smart`.

Value

Returns the usual object, but with one list/slot component called `smart.prediction` containing any data-dependent parameters.

Side Effects

The variables `.max.smart`, `.smart.prediction` and `.smart.prediction.counter` are created while the model is being fitted. In R they are created in a new environment called `smartpredenv`. In S-PLUS they are created in frame 1. These variables are deleted after the model has been fitted. However, in R, if there is an error in the model fitting function or the fitting model is killed (e.g., by typing control-C) then these variables will be left in `smartpredenv`. At the beginning of model fitting, these variables are deleted if present in `smartpredenv`.

During prediction, the variables `.smart.prediction` and `.smart.prediction.counter` are reconstructed and read by the smart functions when the model frame is re-evaluated. After prediction, these variables are deleted.

If the modelling function is used with argument `smart=FALSE` (e.g., `vglm(..., smart=FALSE)`) then smart prediction will not be used, and the results should match with the original R or S-PLUS functions.

WARNING

In S-PLUS, if the `"bigdata"` library is loaded then it is `detach()`'ed. This is done because `scale` cannot be made smart if `"bigdata"` is loaded (it is loaded by default in the Windows version of Splus 8.0, but not in Linux/Unix). The function `search` tells what is currently attached.

In R and S-PLUS the functions `predict.bs` and `predict.ns` are not smart. That is because they operate on objects that contain attributes only and do not have list components or slots. In R the function `predict.poly` is not smart.

Note

In S-PLUS you will need to load in the **smartpred** library with the argument `first=T`, e.g., `library(smartpred, lib="./mys8libs", first=T)`. Here, `mys8libs` is the name of a directory of installed packages. To install the smartpred package in Linux/Unix, type something like `Splus8 INSTALL -l ./mys8libs ./smartpred_0.8-2.tar.gz`.

Author(s)

T. W. Yee and T. J. Hastie

See Also

`get.smart.prediction`, `get.smart`, `put.smart`, `smart.expression`, `smart.mode.is`, `setup.smart`, `wrapup.smart`. Commonly used data-dependent functions include `scale`, `poly`, `bs`, `ns`. In R, the functions `bs` and `ns` are in the **splines** package, and this library is automatically loaded in because it contains compiled code that `bs` and `ns` call.

The website `http://www.stat.auckland.ac.nz/~yee` contains more information such as how to write a smart function, and other technical details.

The functions `vglm`, `vgam`, `rrvglm` and `cqo` in T. W. Yee's **VGAM** package are examples of modelling functions that employ smart prediction.

Examples

```
# Create some data first
n = 20
set.seed(86) # For reproducibility of the random numbers
x = sort(runif(n))
y = sort(runif(n))
## Not run:
if(is.R()) library(splines)   # To get ns() in R
## End(Not run)
```

```
# This will work for R 1.6.0 and later, but fail for S-PLUS
fit = lm(y ~ ns(x, df=5))
## Not run:
plot(x, y)
lines(x, fitted(fit))
newx = seq(0, 1, len=n)
points(newx, predict(fit, data.frame(x=newx)), type="b", col=2, err=-1)
## End(Not run)

# The following fails for R 1.6.x and later but works with smart prediction
## Not run:
fit = lm(y ~ ns(scale(x), df=5))
fit$smart.prediction
plot(x, y)
lines(x, fitted(fit))
newx = seq(0, 1, len=n)
points(newx, predict(fit, data.frame(x=newx)), type="b", col=2, err=-1)
## End(Not run)

# The following requires the VGAM package to be loaded
## Not run:
library(VGAM)
fit = vlm(y ~ ns(scale(x), df=5))
fit@smart.prediction
plot(x, y)
lines(x, fitted(fit))
newx = seq(0, 1, len=n)
points(newx, predict(fit, data.frame(x=newx)), type="b", col=2, err=-1)
## End(Not run)
```

`snorm` *Skew-Normal Distribution*

Description

Density and random generation for the skew-normal distribution.

Usage

```
dsnorm(x, location = 0, scale = 1, shape = 0)
rsnorm(n, location = 0, scale = 1, shape = 0)
```

Arguments

`x`	vector of quantiles.
`n`	number of observations. Must be a single positive integer.
`location`	The location parameter ξ. A vector.
`scale`	The scale parameter ω. A positive vector.
`shape`	The shape parameter. It is called α in `skewnormal1`.

Details

See skewnormal1, which currently only estimates the shape parameter. More generally here, $Z = \xi + \omega Y$ where Y has a standard skew-normal distribution (see skewnormal1), ξ is the location parameter and ω is the scale parameter.

Value

dsnorm gives the density, rsnorm generates random deviates.

Note

The default values of all three parameters corresponds to the skew-normal being the standard normal distribution.

Author(s)

T. W. Yee

References

http://tango.stat.unipd.it/SN.

See Also

skewnormal1.

Examples

```
## Not run:
n = 200   # grid resolution
shape = 7
x = seq(-4, 4, len=n)
plot(x, dsnorm(x, shape=shape), type="l", col="blue", las=1, ylab="")
abline(v=0, h=0, lty="dashed", col="darkgreen")
lines(x, dnorm(x), col="red")
legend(-3.5, 0.6, leg=c(paste("Blue=dsnorm(x, ", shape,")", sep=""),
       "standard normal density"), lty=1, col=c("blue","red"))
## End(Not run)
```

---

sratio *Ordinal Regression with Stopping Ratios*

Description

Fits a stopping ratio logit/probit/cloglog/cauchit/... regression model to an ordered (preferably) factor response.

Usage

```
sratio(link = "logit", earg = list(),
       parallel = FALSE, reverse = FALSE, zero = NULL)
```

Arguments

In the following, the response Y is assumed to be a factor with ordered values $1, 2, \ldots, M+1$, so that M is the number of linear/additive predictors η_j.

link	Link function applied to the M stopping ratio probabilities. See Links for more choices.
earg	List. Extra argument for the link function. See earg in Links for general information.
parallel	A logical, or formula specifying which terms have equal/unequal coefficients.
reverse	Logical. By default, the stopping ratios used are $\eta_j = logit(P[Y = j \mid Y \geq j])$ for $j = 1, \ldots, M$. If reverse is TRUE, then $\eta_j = logit(P[Y = j+1 \mid Y \leq j+1])$ will be used.
zero	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,...,M}. The default value means none are modelled as intercept-only terms.

Details

There are a number of definitions for the *continuation ratio* in the literature. To make life easier, in the **VGAM** package, we use *continuation* ratios (see cratio) and *stopping* ratios. Continuation ratios deal with quantities such as logit(P[Y>j|Y>=j]).

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Warning

No check is made to verify that the response is ordinal; see ordered.

Note

The response should be either a matrix of counts (with row sums that are all positive), or a factor. In both cases, the y slot returned by vglm/vgam/rrvglm is the matrix of counts.

For a nominal (unordered) factor response, the multinomial logit model (multinomial) is more appropriate.

Here is an example of the usage of the parallel argument. If there are covariates x1, x2 and x3, then parallel = TRUE ~ x1 + x2 -1 and parallel = FALSE ~ x3 are equivalent. This would constrain the regression coefficients for x1 and x2 to be equal; those of the intercepts and x3 would be different.

Author(s)

Thomas W. Yee

References

Agresti, A. (2002) *Categorical Data Analysis*, 2nd ed. New York: Wiley.

Simonoff, J. S. (2003) *Analyzing Categorical Data*, New York: Springer-Verlag.

McCullagh, P. and Nelder, J. A. (1989) *Generalized Linear Models*, 2nd ed. London: Chapman & Hall.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

cratio, acat, cumulative, multinomial, pneumo, logit, probit, cloglog, cauchit.

Examples

```
data(pneumo)
pneumo = transform(pneumo, let=log(exposure.time))
(fit = vglm(cbind(normal,mild,severe) ~ let, sratio(parallel=TRUE), pneumo))
coef(fit, matrix=TRUE)
constraints(fit)
predict(fit)
predict(fit, untransform=TRUE)
```

---

studentt *Student t Distribution*

Description

Estimation of the degrees of freedom for a Student t distribution.

Usage

```
studentt(link.df = "loglog", earg=list())
```

Arguments

link.df	Parameter link function for the degrees of freedom ν. See Links for more choices. The default ensures the parameter is greater than unity.
earg	List. Extra argument for the link. See earg in Links for general information.

Details

The density function is

$$f(y) = \frac{\Gamma((\nu+1)/2)}{\sqrt{\nu\pi}\Gamma(\nu/2)} \left(1 + \frac{y^2}{\nu}\right)^{-(\nu+1)/2}$$

for all real y. Then $E(Y) = 0$ if $\nu > 1$ (returned as the fitted values), and $Var(Y) = \nu/(\nu-2)$ for $\nu > 2$. When $\nu = 1$ then the Student t-distribution corresponds to the standard Cauchy distribution. The degrees of freedom is treated as a parameter to be estimated, and as real and not integer.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

A standard normal distribution corresponds to a t distribution with infinite degrees of freedom. Consequently, if the data is close to normal, there may be convergence problems.

Author(s)

T. W. Yee

References

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

Student (1908) The probable error of a mean. *Biometrika*, **6**, 1–25.

See Also

`normal1`, `loglog`, `TDist`.

Examples

```
n = 500
y = rt(n, df=exp(exp(1)))
fit = vglm(y ~ 1, studentt)
coef(fit, matrix=TRUE)
Coef(fit)
```

tikuv *Short-tailed Symmetric Distribution Family Function*

Description

Fits the short-tailed symmetric distribution of Tiku and Vaughan (1999).

Usage

```
tikuv(d, lmean="identity", lsigma="loge",
      emean=list(), esigma=list(),
      isigma=NULL, zero=2)
```

Arguments

d	The d parameter. It must be a single numeric value less than 2. Then $h = 2 - d > 0$ is another parameter.
lmean, lsigma	Link functions for the mean and standard deviation parameters of the usual univariate normal distribution (see **Details** below). They are μ and σ respectively. See Links for more choices.
emean, esigma	List. Extra argument for each of the links. See earg in Links for general information.
isigma	Optional initial value for σ. A NULL means a value is computed internally.
zero	An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2} corresponding respectively to μ, σ. If zero=NULL then all linear/additive predictors are modelled as a linear combination of the explanatory variables. For many data sets having zero=2 is a good idea.

Details

The short-tailed symmetric distribution of Tiku and Vaughan (1999) has a probability density function that can be written

$$f(y) = \frac{K}{\sqrt{2\pi}\sigma}\left[1 + \frac{1}{2h}\left(\frac{y-\mu}{\sigma}\right)^2\right]^2 \exp\left(-\frac{1}{2}(y-\mu)^2/\sigma^2\right)$$

where $h = 2 - d > 0$, K is a function of h, $-\infty < y < \infty$, $\sigma > 0$. The mean of Y is $E(Y) = \mu$ and this is returned as the fitted values.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

Under- or over-flow may occur if the data is ill-conditioned, e.g., when d is very close to 2 or approaches `-Inf`.

Note

The density function is the product of a univariate normal density and a polynomial in the response y. The distribution is bimodal if $d > 0$, else is unimodal. A normal distribution arises as the limit as d approaches $-\infty$, i.e., as h approaches ∞. Fisher scoring is implemented. After fitting the value of `d` is stored as `@misc$d`.

Author(s)

Thomas W. Yee

References

Akkaya, A. D. and Tiku, M. L. (2006) Short-tailed distributions and inliers. *Test*, **15**(2), in press.

Tiku, M. L. and Vaughan, D. C. (1999) A family of short-tailed symmetric distributions. *Technical report, McMaster University, Canada*.

See Also

`dtikuv`, `normal1`.

Examples

```
m = 1.0; sigma = exp(0.5)
sy = sort(y <- rtikuv(n=1000, d=1, m=m, s=sigma))
fit = vglm(y ~ 1, fam=tikuv(d=1), trace=TRUE)
coef(fit, mat=TRUE)
(Cfit = Coef(fit))
mean(y)
## Not run:
hist(y, prob=TRUE)
lines(sy, dtikuv(sy, d=1, m=Cfit[1], s=Cfit[2]), col="red")
## End(Not run)
```

`Tikuv` *A Short-tailed Symmetric Distribution*

Description

Density, cumulative distribution function, quantile function and random generation for the short-tailed symmetric distribution of Tiku and Vaughan (1999).

Usage

```
dtikuv(x, d, mean=0, sigma=1)
ptikuv(q, d, mean=0, sigma=1)
qtikuv(p, d, mean=0, sigma=1, ...)
rtikuv(n, d, mean=0, sigma=1, Smallno=1.0e-6)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. Must be a positive integer of length 1.
`d, mean, sigma`	arguments for the parameters of the distribution. See `tikuv` for more details. For `rtikuv`, arguments `mean` and `sigma` must be of length 1.
`Smallno`	Numeric, a small value used by the rejection method for determining the lower and upper limits of the distribution. That is, `ptikuv(L) < Smallno` and `ptikuv(U) > 1-Smallno` where `L` and `U` are the lower and upper limits respectively.
`...`	Arguments that can be passed into `uniroot`.

Details

See `tikuv` for more details.

Value

`dtikuv` gives the density, `ptikuv` gives the cumulative distribution function, `qtikuv` gives the quantile function, and `rtikuv` generates random deviates.

Author(s)

T. W. Yee

See Also

`tikuv`.

Examples

```
## Not run:
x = seq(-5, 5, len=401)
plot(x, dnorm(x), type="l", col="black", ylab="",
     main="Black is standard normal, others are dtikuv(x, d)")
lines(x, dtikuv(x, d=-10), col="red")
lines(x, dtikuv(x, d=-1 ), col="blue")
lines(x, dtikuv(x, d= 1 ), col="green")
legend(2, 0.35, col=c("red","blue","green"), lty=rep(1, len=3),
       legend=paste("d =", c(-10, -1, 1)))
```

```
plot(x, pnorm(x), type="l", col="black", ylab="",
     main="Black is standard normal, others are ptikuv(x, d)")
lines(x, ptikuv(x, d=-10), col="red")
lines(x, ptikuv(x, d=-1 ), col="blue")
lines(x, ptikuv(x, d= 1 ), col="green")
legend(2, 0.35, col=c("red","blue","green"), lty=rep(1, len=3),
       legend=paste("d =", c(-10, -1, 1)))

probs = seq(0.1, 0.9, by=0.1)
ptikuv(qtikuv(p=probs, d= 1), d=1) - probs  # Should be all 0
## End(Not run)
```

`tobit` *Tobit Model*

Description

Fits a Tobit model to a univariate response.

Usage

```
tobit(Lower = 0, Upper = Inf, lmu="identity", lsd="loge",
      emu=list(), esd=list(), imethod=1, zero=2)
```

Arguments

`Lower`	Numeric of length 1, it is the value L described below. Any value of the linear model $x_i^T\beta$ that is less than this value is assigned this value. Hence this should be the smallest possible value in the response variable.
`Upper`	Numeric of length 1, it is the value U described below. Any value of the linear model $x_i^T\beta$ that is greater than this value is assigned this value. Hence this should be the largest possible value in the response variable.
`lmu, lsd`	Parameter link functions applied to the mean and standard deviation parameters. See `Links` for more choices. The standard deviation is a positive quantity, therefore a log link is its default.
`emu, esd`	List. Extra argument for each of the links. See `earg` in `Links` for general information.
`imethod`	Initialization method. Either 1 or 2, this specifies two methods for obtaining initial values for the parameters.
`zero`	An integer vector, containing the value 1 or 2. If so, the mean or standard deviation respectively are modelled as an intercept only. Setting `zero=NULL` means both linear/additive predictors are modelled as functions of the explanatory variables.

Details

The Tobit model can be written

$$y_i^* = x_i^T \beta + \varepsilon_i$$

where the $e_i \sim N(0, \sigma^2)$ independently and $i = 1, \ldots, n$. However, we measure $y_i = y_i^*$ only if $y_i^* > L$ and $y_i^* < U$ for some cutpoints L and U. Otherwise we let $y_i = L$ or $y_i = U$, whatever is closer. The Tobit model is thus a multiple linear regression but with censored responses if it is below or above certain cutpoints.

The defaults for `Lower` and `Upper` correspond to the classical Tobit model. By default, the mean $x_i^T \beta$ is the first linear/additive predictor, and the log of the standard deviation is the second linear/additive predictor. The Fisher information matrix for uncensored data is diagonal. The fitted values are the estimates of $x_i^T \beta$.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

The response must be univariate. If there is no censoring then `normal1` is recommended instead. Any value of the response less than `Lower` or greater than `Upper` will be assigned the value `Lower` and `Upper` respectively, and a warning will be issued. The fitted object has components `censoredL` and `censoredU` in the `extra` slot which specifies whether observations are censored in that direction.

Author(s)

Thomas W. Yee

References

Tobin, J. (1958) Estimation of relationships for limited dependent variables. *Econometrica* **26**, 24–36.

See Also

`normal1`, `cnormal1`. `dcnormal1`.

Examples

```
n = 1000
x = seq(-1, 1, len=n)
f = function(x) 1 + 4*x
ystar = f(x) + rnorm(n)
Lower = 1
Upper = 4
y  = pmax(ystar, Lower)
y  = pmin(y, Upper)
table(y==Lower | y==Upper)   # How many censored values?
```

```
fit = vglm(y ~ x, tobit(Lower=Lower, Upper=Upper), trace=TRUE)
table(fit@extra$censoredL)
table(fit@extra$censoredU)
coef(fit, matrix=TRUE)
summary(fit)
## Not run:
plot(x, y, main="Tobit model", las=1)
legend(-0.9, 3, c("Truth", "Estimate"), col=c("Blue", "Red"), lwd=2)
lines(x, f(x), col="blue", lwd=2)  # The truth
lines(x, fitted(fit), col="red", lwd=2, lty="dashed")  # The estimate
## End(Not run)
```

`toxop` *Toxoplasmosis Data*

Description

Toxoplasmosis data in 34 cities in El Salvador.

Usage

```
data(toxop)
```

Format

A data frame with 34 observations on the following 4 variables.

`rainfall` a numeric vector; the amount of rainfall in each city.

`ssize` a numeric vector; sample size.

`cityNo` a numeric vector; the city number.

`positive` a numeric vector; the number of subjects testing positive for the disease.

Details

See the references for details.

Source

See the references for details.

References

Efron, B. (1978) Regression and ANOVA With zero-one data: measures of residual variation. *Journal of the American Statistical Association*, **73**, 113–121.

Efron, B. (1986) Double exponential families and their use in generalized linear regression. *Journal of the American Statistical Association*, **81**, 709–721.

See Also

dexpbinomial.

Examples

```
data(toxop)
## Not run:
with(toxop, plot(rainfall, positive/ssize, col="blue"))
plot(toxop, col="blue")
## End(Not run)
```

Tpareto *The Truncated Pareto Distribution*

Description

Density, distribution function, quantile function and random generation for the upper truncated Pareto(I) distribution with parameters `lower`, `upper` and `shape`.

Usage

```
dtpareto(x, lower, upper, shape)
ptpareto(q, lower, upper, shape)
qtpareto(p, lower, upper, shape)
rtpareto(n, lower, upper, shape)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. Must be a single positive integer.
`lower, upper, shape`	the lower, upper and shape (k) parameters. If necessary, values are recycled.

Details

See `tpareto1`, the **VGAM** family function for estimating the parameter k by maximum likelihood estimation, for the formula of the probability density function and the range restrictions imposed on the parameters.

Value

`dtpareto` gives the density, `ptpareto` gives the distribution function, `qtpareto` gives the quantile function, and `rtpareto` generates random deviates.

Author(s)

T. W. Yee

References

Aban, I. B., Meerschaert, M. M. and Panorska, A. K. (2006) Parameter estimation for the truncated Pareto distribution, *Journal of the American Statistical Association*, **101**(473), 270–277.

See Also

tpareto1.

Examples

```
lower = 3; upper = 8; k = exp(0.5)
## Not run:
x = seq(lower, upper, len=200)
plot(x, dtpareto(x, lo=lower, up=upper, shape=k), type="l",
     main="Truncated Pareto density split into 10 equal areas")
abline(h=0, col="blue", lty=2)
qq = qtpareto(seq(0.1,0.9,by=0.1),lo=lower, up=upper,shape=k)
lines(qq, dtpareto(qq, lo=lower, up=upper, shape=k),
      col="purple", lty=3, type="h")
## End(Not run)
pp = seq(0.1,0.9,by=0.1)
qq = qtpareto(pp, lo=lower, up=upper, shape=k)
ptpareto(qq, lo=lower, up=upper, shape=k)
qtpareto(ptpareto(qq, lo=lower, up=upper, shape=k),
         lo=lower, up=upper, shape=k) - qq # Should be all 0
```

---

triangle *Triangle Distribution Family Function*

Description

Estimating the parameter of the triangle distribution by maximum likelihood estimation.

Usage

```
triangle(lower=0, upper=1, link="elogit",
         earg=if(link=="elogit") list(min = lower, max = upper) else
         list(), itheta=NULL)
```

Arguments

lower, upper	lower and upper limits of the distribution. Must be finite. Called A and B respectively below.
link	Parameter link function applied to the parameter θ, which lies in (A, B). See Links for more choices. The default constrains the estimate to lie in the interval.
earg	List. Extra argument for the link. See earg in Links for general information.
itheta	Optional initial value for the parameter. The default is to compute the value internally.

Details

The triangle distribution has a probability density function that consists of two lines joined at θ. The lines intersect the $y = 0$ axis at A and B. Here, Fisher scoring is used.

On fitting, the `extra` slot has components called `lower` and `upper` which contains the values of the above arguments (recycled to the right length). The fitted values are the mean of the distribution, which is a little messy to write.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Note

The response must contain values in (A, B). For most data sets (especially small ones) it is very common for half-stepping to occur.

Author(s)

T. W. Yee

See Also

`Triangle`.

Examples

```
y  = rtriangle(n <- 3000, theta=3/4)
fit = vglm(y ~ 1, triangle(link="identity"), trace=TRUE)
coef(fit, matrix=TRUE)
Coef(fit)
fit@extra$lower[1:5]

fitted(fit)[1:5]
mean(y)
```

`Triangle` *The Triangle Distribution*

Description

Density, distribution function, quantile function and random generation for the Triangle distribution with parameter `theta`.

Usage

```
dtriangle(x, theta, lower=0, upper=1)
ptriangle(q, theta, lower=0, upper=1)
qtriangle(p, theta, lower=0, upper=1)
rtriangle(n, theta, lower=0, upper=1)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. Must be a positive integer of length 1.
`theta`	the theta parameter which lies between `lower` and `upper`.
`lower, upper`	lower and upper limits of the distribution. Must be finite.

Details

See `triangle`, the **VGAM** family function for estimating the parameter θ by maximum likelihood estimation.

Value

`dtriangle` gives the density, `ptriangle` gives the distribution function, `qtriangle` gives the quantile function, and `rtriangle` generates random deviates.

Author(s)

T. W. Yee

See Also

`triangle`.

Examples

```
## Not run:
x = seq(-0.1, 1.1, by=0.01)
theta = 0.75
plot(x, dtriangle(x, theta=theta), type="l", col="blue", las=1,
     main="Blue is density, red is cumulative distribution function",
     sub="Purple lines are the 10,20,...,90 percentiles",
     ylim=c(0,2), ylab="")
abline(h=0, col="blue", lty=2)
lines(x, ptriangle(x, theta=theta), col="red")
probs = seq(0.1, 0.9, by=0.1)
Q = qtriangle(probs, theta=theta)
lines(Q, dtriangle(Q, theta=theta), col="purple", lty=3, type="h")
ptriangle(Q, theta=theta) - probs    # Should be all zero
abline(h=probs, col="purple", lty=3)
## End(Not run)
```

trplot *Trajectory Plot*

Description

Generic function for a trajectory plot.

Usage

```
trplot(object, ...)
```

Arguments

object	An object for which a trajectory plot is meaningful.
...	Other arguments fed into the specific methods function of the model. They usually are graphical parameters, and sometimes they are fed into the methods function for Coef.

Details

Trajectory plots can be defined in different ways for different models. Many models have no such notion or definition.

For quadratic and additive ordination models they plot the fitted values of two species against each other (more than two is theoretically possible, but not implemented in this software yet).

Value

The value returned depends specifically on the methods function invoked.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2005) On constrained and unconstrained quadratic ordination. *Manuscript in preparation*.

See Also

trplot.qrrvglm, persp.qrrvglm, lvplot.

Examples

```
## Not run:
data(hspider)
hspider[,1:6] = scale(hspider[,1:6]) # Standardized environmental vars
set.seed(123)

p1 = cqo(cbind(Alopacce, Alopcune, Alopfabr, Arctlute, Arctperi,
               Auloalbi, Pardlugu, Pardmont, Pardnigr, Pardpull,
               Trocterr, Zoraspin) ~
         WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + ReflLux,
         fam = quasipoissonff, data = hspider, Crow1positive=FALSE)

nos = ncol(p1@y)
clr = 1:nos   # OR (1:(nos+1))[-7]  to omit yellow

trplot(p1, whichSpecies=1:3, log="xy",
       col=c("blue","red","green"), lwd=2, label=TRUE) -> ii
legend(0.00005, 0.3, paste(ii$species[,1], ii$species[,2], sep=" and "),
       lwd=2, lty=1, col=c("blue","red","green"))
abline(a=0, b=1, lty="dashed")

## End(Not run)
```

---

trplot.qrrvglm *Trajectory plot for QRR-VGLMs*

Description

Produces a trajectory plot for *quadratic reduced-rank vector generalized linear models* (QRR-VGLMs). It is only applicable for rank-1 models with argument `Norrr = ~ 1`.

Usage

```
trplot.qrrvglm(object, whichSpecies = NULL,
               add=FALSE, plot.it=TRUE,
               label.sites = FALSE,
               sitenames = dimnames(object@y)[[1]],
               axes.equal = TRUE, cex = par()$cex,
               col = 1:(nos * (nos - 1)/2), log = "",
               lty = rep(par()$lty, len = nos * (nos - 1)/2),
               lwd = rep(par()$lwd, len = nos * (nos - 1)/2),
               tcol = rep(par()$col, len = nos * (nos - 1)/2),
               xlab = NULL, ylab = NULL,
               main = "", type = "b", check.ok=TRUE, ...)
```

Arguments

object	Object of class "qrrvglm", i.e., a CQO object.
whichSpecies	Integer or character vector specifying the species to be plotted. If integer, these are the columns of the response matrix. If character, these must match exactly with the species' names. The default is to use all species.
add	Logical. Add to an existing plot? If FALSE (default), a new plot is made.
plot.it	Logical. Plot it?
label.sites	Logical. If TRUE, the points on the curves/trajectories are labelled with the sitenames.
sitenames	Character vector. The names of the sites.
axes.equal	Logical. If TRUE, the x- and y-axes will be on the same scale.
cex	Character expansion of the labelling of the site names. Used only if label.sites is TRUE. See the cex argument in par.
col	Color of the lines. See the col argument in par. Here, nos is the number of species.
log	Character, specifying which (if any) of the x- and y-axes are to be on a logarithmic scale. See the log argument in par.
lty	Line type. See the lty argument of par.
lwd	Line width. See the lwd argument of par.
tcol	Color of the text for the site names. See the col argument in par. Used only if label.sites is TRUE.
xlab	Character caption for the x-axis. By default, a suitable caption is found. See the xlab argument in plot or title.
ylab	Character caption for the y-axis. By default, a suitable caption is found. See the xlab argument in plot or title.
main	Character, giving the title of the plot. See the main argument in plot or title.
type	Character, giving the type of plot. A common option is to use type="l" for lines only. See the type argument of plot.
check.ok	Logical. Whether a check is performed to see that Norrr = ~ 1 was used. It doesn't make sense to have a trace plot unless this is so.
...	Arguments passed into the plot function when setting up the entire plot. Useful arguments here include xlim and ylim.

Details

A trajectory plot plots the fitted values of a 'second' species against a 'first' species. The argument whichSpecies must therefore contain at least two species. By default, all of the species that were fitted in object are plotted. With more than a few species the resulting plot will be very congested, and so it is recommended that only a few species be selected for plotting.

In the above, M is the number of species selected for plotting, so there will be $M(M-1)/2$ curves/trajectories in total.

A trajectory plot will be fitted only if Norrr = ~ 1 because otherwise the trajectory will not be a smooth function of the latent variables.

Value

A list with the following components.

`species.names`
: A matrix of characters giving the 'first' and 'second' species. The number of different combinations of species is given by the number of rows. This is useful for creating a legend.

`sitenames`
: A character vector of site names, sorted by the latent variable (from low to high).

Note

Plotting the axes on a log scale is often a good idea. The use of `xlim` and `ylim` to control the axis limits is also a good idea, so as to limit the extent of the curves at low abundances or probabilities. Setting `label.sites=TRUE` is a good idea only if the number of sites is small, otherwise there is too much clutter.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2005) On constrained and unconstrained quadratic ordination. *Manuscript in preparation*.

See Also

`cqo`, `par`, `title`.

Examples

```
## Not run:
data(hspider)
set.seed(111)  # This leads to the global solution
# hspider[,1:6]=scale(hspider[,1:6]) # Standardize the environmental variables
p1 = cqo(cbind(Alopacce, Alopcune, Alopfabr, Arctlute, Arctperi, Auloalbi,
               Pardlugu, Pardmont, Pardnigr, Pardpull, Trocterr, Zoraspin) ~
         WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + ReflLux,
         fam = poissonff, data = hspider, trace=FALSE)

trplot(p1, whichSpecies=1:3, log="xy", type="b", lty=1,
       main="Trajectory plot of three hunting spiders species",
       col=c("blue","red","green"), lwd=2, label=TRUE) -> ii
legend(0.00005, 0.3, paste(ii$species[,1], ii$species[,2], sep=" and "),
       lwd=2, lty=1, col=c("blue","red","green"))
abline(a=0, b=1, lty="dashed")  # Useful reference line
## End(Not run)
```

undocumented-methods *Undocumented Methods Functions*

Description

Lots of undocumented methods functions are aliased here. In the **VGAM** package there are currently many objects/methods/classes which are currently internal and/or undocumented. The help file suppresses the warnings when the package is 'CHECK'ed.

Methods

There are many methods and these will be documented over time.

object This argument is often used, and it is the primary object from which the function operates on.

uqo *Fitting Unconstrained Quadratic Ordination (UQO)*

Description

An *unconstrained quadratic ordination* (UQO) (equivalently, noncanonical Gaussian ordination) model is fitted using the *quadratic unconstrained vector generalized linear model* (QU-VGLM) framework. In this documentation, M is the number of linear predictors or species.

Usage

```
uqo(formula, family, data = list(), weights = NULL, subset = NULL,
    na.action = na.fail, etastart = NULL, mustart = NULL,
    coefstart = NULL, control = uqo.control(...), offset = NULL,
    method = "uqo.fit", model = FALSE, x.arg = TRUE, y.arg = TRUE,
    contrasts = NULL, constraints = NULL, extra = NULL,
    qr.arg = FALSE, ...)
```

Arguments

`formula`	a symbolic description of the model to be fit. Since there is no x_2 vector by definition, the RHS of the formula has all terms belonging to the x_1 vector.
`family`	a function of class `"vglmff"` describing what statistical model is to be fitted. Currently two families are supported: Poisson and binomial.
`data`	an optional data frame containing the variables in the model. By default the variables are taken from `environment(formula)`, typically the environment from which `uqo` is called.

weights	an optional vector or matrix of (prior) weights to be used in the fitting process. This argument should not be used.
subset	an optional logical vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The "factory-fresh" default is na.omit.
etastart	starting values for the linear predictors. It is a M-column matrix. If $M = 1$ then it may be a vector.
mustart	starting values for the fitted values. It can be a vector or a matrix. Some family functions do not make use of this argument.
coefstart	starting values for the coefficient vector.
control	a list of parameters for controlling the fitting process. See uqo.control for details.
offset	a vector or M-column matrix of offset values. This argument should not be used.
method	the method to be used in fitting the model. The default (and presently only) method uqo.fit uses iteratively reweighted least squares (IRLS).
model	a logical value indicating whether the *model frame* should be assigned in the model slot.
x.arg, y.arg	logical values indicating whether the model matrix and response matrix used in the fitting process should be assigned in the x and y slots. Note the model matrix is the LM model matrix.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
constraints	an optional list of constraint matrices. This argument should not be used.
extra	an optional list with any extra information that might be needed by the family function.
qr.arg	logical value indicating whether the slot qr, which returns the QR decomposition of the VLM model matrix, is returned on the object. This argument should not be set TRUE.
...	further arguments passed into uqo.control.

Details

Unconstrained quadratic ordination models fit symmetric bell-shaped response curves/surfaces to response data, but the latent variables are largely free parameters and are not constrained to be linear combinations of the environmental variables. This poses a difficult optimization problem. The current algorithm is very simple and will often fail (even for Rank=1) but hopefully this will be improved in the future.

The central formula is given by

$$\eta = B_1^T x_1 + A\nu + \sum_{m=1}^{M} (\nu^T D_m \nu) e_m$$

where x_1 is a vector (usually just a 1 for an intercept), ν is a R-vector of latent variables, e_m is a vector of 0s but with a 1 in the mth position. The η are a vector of linear/additive predictors, e.g., the mth element is $\eta_m = \log(E[Y_m])$ for the mth species. The matrices B_1, A, and D_m are estimated from the data, i.e., contain the regression coefficients. Also, ν is estimated. The tolerance matrices satisfy $T_s = -\frac{1}{2} D_s^{-1}$. Many important UQO details are directly related to arguments in `uqo.control`; see also `cqo` and `qrrvglm.control`.

Currently, only Poisson and binomial **VGAM** family functions are implemented for this function, and dispersion parameters for these are assumed known. Thus the Poisson is catered for by `poissonff`, and the binomial by `binomialff`. Those beginning with `"quasi"` have dispersion parameters that are estimated for each species, hence will give an error message here.

Value

An object of class `"uqo"` (this may change to `"quvglm"` in the future).

Warning

Local solutions are not uncommon when fitting UQO models. To increase the chances of obtaining the global solution, set `ITolerances=TRUE` or `EqualTolerances=TRUE` and increase the value of the argument `Bestof` in `uqo.control`. For reproducibility of the results, it pays to set a different random number seed before calling `uqo` (the function `set.seed` does this).

The function `uqo` is very sensitive to initial values, and there is a lot of room for improvement here.

UQO is computationally expensive. It pays to keep the rank to no more than 2, and 1 is much preferred over 2. The data needs to conform closely to the statistical model.

Currently there is a bug with the argument `Crow1positive` in `uqo.control`. This argument might be interpreted as controlling the sign of the first site score, but currently this is not done.

Note

The site scores are centered. When $R > 1$, they are uncorrelated and should be unique up to a rotation.

The argument `Bestof` in `uqo.control` controls the number of models fitted (each uses different starting values) to the data. This argument is important because convergence may be to a *local* solution rather than the *global* solution. Using more starting values increases the chances of finding the global solution. Local solutions arise because the optimization problem is highly nonlinear.

In the example below, a CQO model is fitted and used for providing initial values for a UQO model.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2004) A new technique for maximum-likelihood canonical Gaussian ordination. *Ecological Monographs*, **74**, 685–701.

Yee, T. W. (2005) On constrained and unconstrained quadratic ordination. *Manuscript in preparation*.

Yee, T. W. (2006) Constrained additive ordination. *Ecology*, **87**, 203–213.

See Also

uqo.control, cqo, qrrvglm.control, rcqo, poissonff, binomialff, Coef.uqo, lvplot.uqo, persp.uqo, trplot.uqo, vcov.uqo, set.seed, hspider.

Examples

```
## Not run:
data(hspider)
set.seed(123)  # This leads to the global solution
hspider[,1:6] = scale(hspider[,1:6]) # Standardized environmental vars
p1 = cqo(cbind(Alopacce, Alopcune, Alopfabr, Arctlute, Arctperi,
               Auloalbi, Pardlugu, Pardmont, Pardnigr, Pardpull,
               Trocterr, Zoraspin) ~
         WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + ReflLux,
         ITolerances = TRUE, fam = poissonff, data = hspider,
         Crow1positive=TRUE, Bestof=3, trace=FALSE)
if(deviance(p1) > 1589.0) stop("suboptimal fit obtained")

set.seed(111)
up1 = uqo(cbind(Alopacce, Alopcune, Alopfabr, Arctlute, Arctperi,
                Auloalbi, Pardlugu, Pardmont, Pardnigr, Pardpull,
                Trocterr, Zoraspin) ~ 1,
          family = poissonff, data = hspider,
          ITolerances = TRUE,
          Crow1positive = TRUE, lvstart = -lv(p1))
if(deviance(up1) > 1310.0) stop("suboptimal fit obtained")

nos = ncol(up1@y) # Number of species
clr = (1:(nos+1))[-7]  # to omit yellow
lvplot(up1, las=1, y=TRUE, pch=1:nos, scol=clr, lcol=clr,
       pcol=clr, llty=1:nos, llwd=2)
legend(x=2, y=135, dimnames(up1@y)[[2]], col=clr, lty=1:nos,
       lwd=2, merge=FALSE, ncol=1, x.inter=4.0, bty="l", cex=0.9)

# Compare the site scores between the two models
plot(lv(p1), lv(up1), xlim=c(-3,4), ylim=c(-3,4), las=1)
abline(a=0, b=-1, lty=2, col="blue", xpd=FALSE)
cor(lv(p1, ITol=TRUE), lv(up1))

# Another comparison between the constrained and unconstrained models
# The signs are not right so they are similar when reflected about 0
par(mfrow=c(2,1))
persp(up1, main="Red/Blue are the constrained/unconstrained models",
      label=TRUE, col="blue", las=1)
persp(p1, add=FALSE, col="red")
1-pchisq(deviance(p1) - deviance(up1), df=52-30)
## End(Not run)
```

---

uqo.control *Control Function for UQO models*

Description

Algorithmic constants and parameters for an unconstrained quadratic ordination (UQO) model, by fitting a *quadratic unconstrained vector generalized additive model* (QU-VGLM), are set using this function. It is the control function of `uqo`.

Usage

```
uqo.control(Rank=1, Bestof = if(length(lvstart) &&
            !jitter.sitescores) 1 else 10, CA1 = FALSE, Crow1positive
            = TRUE, epsilon = 1.0e-07, EqualTolerances = ITolerances,
            Etamat.colmax = 10, GradientFunction=TRUE, Hstep = 0.001,
            isdlv = rep(c(2, 1, rep(0.5, len=Rank)), len=Rank),
            ITolerances = FALSE, lvstart = NULL, jitter.sitescores
            = FALSE, maxitl = 40, Maxit.optim = 250, MUXfactor =
            rep(3, length=Rank), optim.maxit = 20, nRmax = 250,
            SD.sitescores = 1.0, SmallNo = 5.0e-13, trace = TRUE,
            Use.Init.Poisson.QO=TRUE, ...)
```

Arguments

`Rank` The numerical rank R of the model, i.e., the number of latent variables or ordination axes. Currently only $R = 1$ is recommended.

`Bestof` Integer. The best of `Bestof` models fitted is returned. This argument helps guard against local solutions by (hopefully) finding the global solution from many fits. The argument has value 1 if an initial value for the site scores is inputted using `lvstart`.

`CA1` Logical. If `TRUE` the site scores from a correspondence analysis (CA) are computed and used on the first axis as initial values. Both `CA1` and `Use.Init.Poisson.QO` cannot both be `TRUE`.

`Crow1positive` Logical vector of length `Rank` (recycled if necessary): are the elements of the first row of the latent variable matrix ν positive? For example, if `Rank` is 2, then specifying `Crow1positive=c(FALSE, TRUE)` will force the first site score's first element to be negative, and the first site score's second element to be positive. Note that there is no C matrix with UQO, but the argument's name comes from `qrrvglm.control` and is left unchanged for convenience.

`epsilon` Positive numeric. Used to test for convergence for GLMs fitted in FORTRAN. Larger values mean a loosening of the convergence criterion.

`EqualTolerances` Logical indicating whether each (quadratic) predictor will have equal tolerances. Setting `EqualTolerances=TRUE` can help avoid numerical problems, especially with binary data. Note that the estimated (common) tolerance matrix may or may not be positive-definite. If it is, then it can be scaled to the R x R identity matrix. Setting `ITolerances=TRUE` will fit a common R x R identity matrix as the tolerance matrix to the data, but this is model-driven rather than being data-driven because it *forces* bell-shaped curves/surfaces onto the data. If the estimated (common) tolerance matrix happens to be positive-definite, then

this model is essentially equivalent to the model with ITolerances=TRUE. See **Details** in cqo and qrrvglm.control for more details.

Etamat.colmax
: Positive integer, no smaller than Rank. Controls the amount of memory used by .Init.Poisson.QO(). It is the maximum number of columns allowed for the pseudo-response and its weights. In general, the larger the value, the better the initial value. Used only if Use.Init.Poisson.QO=TRUE.

GradientFunction
: Logical. Whether optim's argument gr is used or not, i.e., to compute gradient values. The default value is usually faster on most problems.

Hstep
: Positive value. Used as the step size in the finite difference approximation to the derivatives by optim.

isdlv
: Initial standard deviations for the latent variables (site scores). Numeric, positive and of length R (recycled if necessary). This argument is used only if ITolerances=TRUE. Used by .Init.Poisson.QO() to obtain initial values for the constrained coefficients C adjusted to a reasonable value. It adjusts the spread of the site scores relative to a common species tolerance of 1 for each ordination axis. A value between 0.5 and 10 is recommended; a value such as 10 means that the range of the environmental space is very large relative to the niche width of the species. The successive values should decrease because the first ordination axis should have the most spread of site scores, followed by the second ordination axis, etc.

ITolerances
: Logical. If TRUE then the (common) tolerance matrix is the R x R identity matrix by definition. Note that ITolerances=TRUE implies EqualTolerances=TRUE, but not vice versa. Internally, the quadratic terms will be treated as offsets (in GLM jargon) and so the models can potentially be fitted very efficiently. See **Details** in cqo and qrrvglm.control for more details. more details. The success of ITolerances=TRUE often depends on suitable values for isdlv and/or MUXfactor.

lvstart
: Optional matrix of initial values of the site scores. If given, the matrix must be n by R, where n is the number of sites and R is the rank. This argument overrides the arguments Use.Init.Poisson.QO and CA1. Good possibilities for lvstart are the site scores from a constrained ordination, e.g., from cqo.

jitter.sitescores
: Logical. If TRUE the initial values for the site scores are jittered to add a random element to the starting values.

maxitl
: Positive integer. Number of iterations allowed for the IRLS algorithm implemented in the compiled code.

Maxit.optim
: Positive integer. Number of iterations given to the function optim at each of the optim.maxit iterations.

MUXfactor
: Multiplication factor for detecting large offset values. Numeric, positive and of length R (recycled if necessary). This argument is used only if ITolerances=TRUE. Offsets are -0.5 multiplied by the sum of the squares of all R latent variable values. If the latent variable values are too large then this will result in numerical problems. By too large, it is meant that the standard deviation of the latent variable values are greater than MUXfactor[r] * isdlv[r] for r=1:Rank

(this is why centering and scaling all the numerical predictor variables in x_2 is recommended). A value about 3 or 4 is recommended. If failure to converge occurs, try a slightly lower value.

`optim.maxit` Positive integer. Number of times `optim` is invoked.

`nRmax` Positive integer. If the number of parameters making up the latent variable values (n multiplied by R) is greater than this argument then a conjugate-gradients algorithm is used, otherwise a quasi-Newton algorithm is used by `optim`. The conjugate-gradients method is more suitable when the number of parameters is large because it requires less memory.

`SD.sitescores`
Numeric. Standard deviation of the initial values of the site scores, which are generated from a normal distribution.

`SmallNo` Positive numeric between `.Machine$double.eps` and `0.0001`. Used to avoid under- or over-flow in the IRLS algorithm.

`trace` Logical indicating if output should be produced for each iteration.

`Use.Init.Poisson.QO`
Logical. If `TRUE` then the function `.Init.Poisson.QO()` is used to obtain initial values for the site scores. If `FALSE` then random numbers are used instead. Both `CA1` and `Use.Init.Poisson.QO` cannot both be `TRUE`.

`...` Ignored at present.

Details

The algorithm currently used by `uqo` is unsophisticated and fails often. Improvements will hopefully be made soon.

See `cqo` and `qrrvglm.control` for more details that are equally pertinent to UQO.

To reduce the number of parameters being estimated, setting `ITolerances = TRUE` or `EqualTolerances = TRUE` is advised.

Value

A list with the components corresponding to its arguments, after some basic error checking.

Warning

This function is currently very sensitive to initial values. Setting `Bestof` some reasonably large integer is recommended.

Note

This is a difficult optimization problem, and the current algorithm needs to be improved.

Author(s)

T. W. Yee

References

Yee, T. W. (2005) On constrained and unconstrained quadratic ordination. *Manuscript in preparation*.

Yee, T. W. (2006) Constrained additive ordination. *Ecology*, **87**, 203–213.

See Also

uqo.

Examples

```
uqo.control()
```

`usagrain` *USA grain prices*

Description

A 4-column matrix.

Usage

```
data(usagrain)
```

Format

The columns are:

wheat.flour numeric

corn numeric

wheat numeric

rye numeric

Details

Monthly averages of grain prices in the United States for wheat flour, corn, wheat, and rye for the period January 1961 through October 1972. The units are US dollars per 100 pound sack for wheat flour, and per bushel for corn, wheat and rye.

Source

Ahn and Reinsel (1988)

References

Ahn, S. K and Reinsel, G. C. (1988) Nested reduced-rank autoregressive models for multiple time series. *Journal of the American Statistical Association*, **83**, 849–856.

Examples

```
data(usagrain)
cgrain = scale(usagrain, scale=FALSE)   # Center the time series only
fit = vglm(cgrain ~ 1, rrar(Rank=c(4,1)),
           eps=1e-3, step=0.5, trace=TRUE, maxit=40)
summary(fit)
```

---

venice *Venice Maximum Sea Levels*

Description

The maximum heights of sea levels recorded at Venice, Italy, between 1931 and 1981.

Usage

```
data(venice)
```

Format

A data frame with 51 observations on the following 11 variables.

year a numeric vector.

r1 a numeric vector; the highest recorded value.

r2,r3,r4,r5,r6,r7,r8,r9,r10 numeric vectors; the second highest recorded value down to the tenth highest recorded value.

Details

The units are cm. In 1935 only the top six values were recorded.

Source

Pirazzoli, P. (1982) Maree estreme a Venezia (periodo 1872–1981). *Acqua Aria*, **10**, 1023–1039.

References

Smith, R. L. (1986) Extreme value theory based on the *r* largest annual events. *Journal of Hydrology*, **86**, 27–43.

Rosen, O. and Cohen, A. (1996) Extreme Percentile Regression. In: Haerdle, W. and Schimek, M. G. (eds.) *Statistical Theory and Computational Aspects of Smoothing: Proceedings of the COMPSTAT '94 Satellite Meeting held in Semmering, Austria, 27–28 August 1994*, Physica-Verlag: Heidelberg, pp.200–214.

Examples

```
data(venice)
## Not run:
matplot(venice[["year"]], venice[,-1], xlab="Year", ylab="Sea level (cm)")
## End(Not run)

y = as.matrix(venice[,paste("r",1:10,sep="")])
fit1 = vgam(y ~ s(year, df=3), gumbel(R=365, mpv=TRUE),
            data=venice, trace=TRUE, na.action=na.pass)
fitted(fit1)[1:4,]

## Not run:
par(mfrow=c(2,1), xpd=TRUE)
plot(fit1, se=TRUE, lcol="blue", llwd=2, slty="dashed")

par(mfrow=c(1,1), bty="l", xpd=TRUE, las=1)
qtplot(fit1, mpv=TRUE, lcol=c(1,2,5), tcol=c(1,2,5),
       llwd=2, pcol="blue", tadj=0.1)
## End(Not run)
```

---

vgam-class *Class "vgam"*

Description

Vector generalized additive models.

Objects from the Class

Objects can be created by calls of the form `vgam(...)`.

Slots

`nl.chisq`: Object of class `"numeric"`. Nonlinear chi-squared values.

`nl.df`: Object of class `"numeric"`. Nonlinear chi-squared degrees of freedom values.

`spar`: Object of class `"numeric"` containing the (scaled) smoothing parameters.

`s.xargument`: Object of class `"character"` holding the variable name of any `s()` terms.

`var`: Object of class `"matrix"` holding approximate pointwise standard error information.

`Bspline`: Object of class `"list"` holding the scaled (internal and boundary) knots, and the fitted B-spline coefficients. These are used for prediction.

`extra`: Object of class `"list"`; the `extra` argument on entry to `vglm`. This contains any extra information that might be needed by the family function.

`family`: Object of class `"vglmff"`. The family function.

`iter`: Object of class `"numeric"`. The number of IRLS iterations used.

`predictors`: Object of class `"matrix"` with M columns which holds the M linear predictors.

assign: Object of class "list", from class "vlm". This named list gives information matching the columns and the (LM) model matrix terms.

call: Object of class "call", from class "vlm". The matched call.

coefficients: Object of class "numeric", from class "vlm". A named vector of coefficients.

constraints: Object of class "list", from class "vlm". A named list of constraint matrices used in the fitting.

contrasts: Object of class "list", from class "vlm". The contrasts used (if any).

control: Object of class "list", from class "vlm". A list of parameters for controlling the fitting process. See vglm.control for details.

criterion: Object of class "list", from class "vlm". List of convergence criterion evaluated at the final IRLS iteration.

df.residual: Object of class "numeric", from class "vlm". The residual degrees of freedom.

df.total: Object of class "numeric", from class "vlm". The total degrees of freedom.

dispersion: Object of class "numeric", from class "vlm". The scaling parameter.

effects: Object of class "numeric", from class "vlm". The effects.

fitted.values: Object of class "matrix", from class "vlm". The fitted values. This may be missing or consist entirely of NAs, e.g., the Cauchy model.

misc: Object of class "list", from class "vlm". A named list to hold miscellaneous parameters.

model: Object of class "data.frame", from class "vlm". The model frame.

na.action: Object of class "list", from class "vlm". A list holding information about missing values.

offset: Object of class "matrix", from class "vlm". If non-zero, a M-column matrix of offsets.

post: Object of class "list", from class "vlm" where post-analysis results may be put.

preplot: Object of class "list", from class "vlm" used by plotvgam; the plotting parameters may be put here.

prior.weights: Object of class "numeric", from class "vlm" holding the initially supplied weights.

qr: Object of class "list", from class "vlm". QR decomposition at the final iteration.

R: Object of class "matrix", from class "vlm". The **R** matrix in the QR decomposition used in the fitting.

rank: Object of class "integer", from class "vlm". Numerical rank of the fitted model.

residuals: Object of class "matrix", from class "vlm". The *working* residuals at the final IRLS iteration.

rss: Object of class "numeric", from class "vlm". Residual sum of squares at the final IRLS iteration with the adjusted dependent vectors and weight matrices.

smart.prediction: Object of class "list", from class "vlm". A list of data-dependent parameters (if any) that are used by smart prediction.

terms: Object of class "list", from class "vlm". The terms object used.

weights: Object of class "matrix", from class "vlm". The weight matrices at the final IRLS iteration. This is in matrix-band form.

x: Object of class "matrix", from class "vlm". The model matrix (LM, not VGLM).

xlevels: Object of class "list", from class "vlm". The levels of the factors, if any, used in fitting.

y: Object of class "matrix", from class "vlm". The response, in matrix form.

Extends

Class "vglm", directly. Class "vlm", by class "vglm".

Methods

cdf signature(object = "vglm"): cumulative distribution function. Useful for quantile regression and extreme value data models.

deplot signature(object = "vglm"): density plot. Useful for quantile regression models.

deviance signature(object = "vglm"): deviance of the model (where applicable).

plot signature(x = "vglm"): diagnostic plots.

predict signature(object = "vglm"): extract the additive predictors or predict the additive predictors at a new data frame.

print signature(x = "vglm"): short summary of the object.

qtplot signature(object = "vglm"): quantile plot (only applicable to some models).

resid signature(object = "vglm"): residuals. There are various types of these.

residuals signature(object = "vglm"): residuals. Shorthand for resid.

rlplot signature(object = "vglm"): return level plot. Useful for extreme value data models.

summary signature(object = "vglm"): a more detailed summary of the object.

Note

VGAMs have all the slots that vglm objects have (vglm-class), plus the first few slots described in the section above.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Wild, C. J. (1996) Vector generalized additive models. *Journal of the Royal Statistical Society, Series B, Methodological*, **58**, 481–493.

http://www.stat.auckland.ac.nz/~yee

See Also

vgam.control, vglm, s, vglm-class, vglmff-class.

Examples

```
# Fit a nonparametric proportional odds model
data(pneumo)
pneumo = transform(pneumo, let=log(exposure.time))
vgam(cbind(normal, mild, severe) ~ s(let),
     cumulative(parallel=TRUE), pneumo)
```

---

vgam *Fitting Vector Generalized Additive Models*

Description

Fit a vector generalized additive model (VGAM). This is a large class of models that includes generalized additive models (GAMs) and vector generalized linear models (VGLMs) as special cases.

Usage

```
vgam(formula, family, data = list(), weights = NULL, subset = NULL,
     na.action = na.fail, etastart = NULL, mustart = NULL,
     coefstart = NULL, control = vgam.control(...), offset = NULL,
     method = "vgam.fit", model = FALSE, x.arg = TRUE, y.arg = TRUE,
     contrasts = NULL, constraints = NULL,
     extra = list(), qr.arg = FALSE, smart = TRUE, ...)
```

Arguments

In the following, M is the number of additive predictors.

formula	a symbolic description of the model to be fit. The RHS of the formula is applied to each linear/additive predictor. Different variables in each linear/additive predictor can be chosen by specifying constraint matrices.
family	a function of class "vglmff" (see vglmff-class) describing what statistical model is to be fitted. These are called "**VGAM** family functions".
data	an optional data frame containing the variables in the model. By default the variables are taken from environment(formula), typically the environment from which vgam is called.
weights	an optional vector or matrix of (prior) weights to be used in the fitting process. If weights is a matrix, then it must be in *matrix-band* form, whereby the first M columns of the matrix are the diagonals, followed by the upper-diagonal band, followed by the band above that, etc. In this case, there can be up to $M(M+1)$ columns, with the last column corresponding to the (1,M) elements of the weight matrices.

subset	an optional logical vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The "factory-fresh" default is na.omit.
etastart	starting values for the linear/additive predictors. It is a M-column matrix. If $M = 1$ then it may be a vector.
mustart	starting values for the fitted values. It can be a vector or a matrix. Some family functions do not make use of this argument.
coefstart	starting values for the coefficient vector.
control	a list of parameters for controlling the fitting process. See vgam.control for details.
offset	a vector or M-column matrix of offset values. These are *a priori* known and are added to the linear/additive predictors during fitting.
method	the method to be used in fitting the model. The default (and presently only) method vgam.fit uses iteratively reweighted least squares (IRLS).
model	a logical value indicating whether the *model frame* should be assigned in the model slot.
x.arg, y.arg	logical values indicating whether the model matrix and response vector/matrix used in the fitting process should be assigned in the x and y slots. Note the model matrix is the LM model matrix; to get the VGAM model matrix type model.matrix(vgamfit) where vgamfit is a vgam object.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
constraints	an optional list of constraint matrices. The components of the list must be named with the term it corresponds to (and it must match in character format exactly). Each constraint matrix must have M rows, and be of full-column rank. By default, constraint matrices are the M by M identity matrix unless arguments in the family function itself override these values. If constraints is used it must contain *all* the terms; an incomplete list is not accepted.
extra	an optional list with any extra information that might be needed by the **VGAM** family function.
qr.arg	logical value indicating whether the slot qr, which returns the QR decomposition of the VLM model matrix, is returned on the object.
smart	logical value indicating whether smart prediction (smartpred) will be used.
...	further arguments passed into vgam.control.

Details

A vector generalized additive model (VGAM) is loosely defined as a statistical model that is a function of M additive predictors. The central formula is given by

$$\eta_j = \sum_{k=1}^{p} f_{(j)k}(x_k)$$

where x_k is the kth explanatory variable (almost always $x_1 = 1$ for the intercept term), and $f_{(j)k}$ are smooth functions of x_k that are estimated by smoothers. The first term in the summation is just the intercept. Currently only one type of smoother is implemented and this is called a *vector (cubic smoothing spline) smoother*. Here, $j = 1, \ldots, M$ where M is finite. If all the functions are constrained to be linear then the resulting model is a vector generalized linear model (VGLM). VGLMs are best fitted with `vglm`.

Vector (cubic smoothing spline) smoothers are represented by `s()` (see `s`). Local regression via `lo()` is *not* supported. The results of `vgam` will differ from the S-PLUS and R `gam` function (in the **gam** R package) because `vgam` uses a different knot selection algorithm. In general, fewer knots are chosen because the computation becomes expensive when the number of additive predictors M is large.

The underlying algorithm of VGAMs is iteratively reweighted least squares (IRLS) and modified vector backfitting using vector splines. B-splines are used as the basis functions for the vector (smoothing) splines. `vgam.fit` is the function that actually does the work. The smoothing code is based on F. O'Sullivan's BART code.

A closely related methodology based on VGAMs called *constrained additive ordination* (CAO) first forms a linear combination of the explanatory variables (called *latent variables*) and then fits a GAM to these. This is implemented in the function `cao` for a very limited choice of family functions.

Value

An object of class `"vgam"` (see `vgam-class` for further information).

Note

This function can fit a wide variety of statistical models. Some of these are harder to fit than others because of inherent numerical difficulties associated with some of them. Successful model fitting benefits from cumulative experience. Varying the values of arguments in the **VGAM** family function itself is a good first step if difficulties arise, especially if initial values can be inputted. A second, more general step, is to vary the values of arguments in `vgam.control`. A third step is to make use of arguments such as `etastart`, `coefstart` and `mustart`.

Some **VGAM** family functions end in `"ff"` to avoid interference with other functions, e.g., `binomialff`, `poissonff`, `gaussianff`, `gammaff`. This is because **VGAM** family functions are incompatible with `glm` (and also `gam` in the **gam** library and `gam` in the **mgcv** library).

The smart prediction (`smartpred`) library is packed with the **VGAM** library.

The theory behind the scaling parameter is currently being made more rigorous, but it it should give the same value as the scale parameter for GLMs.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Wild, C. J. (1996) Vector generalized additive models. *Journal of the Royal Statistical Society, Series B, Methodological*, **58**, 481–493.

http://www.stat.auckland.ac.nz/~yee

See Also

vgam.control, vgam-class, vglmff-class, plotvgam, vglm, s, vsmooth.spline, cao.

Examples

```
# Nonparametric proportional odds model
data(pneumo)
pneumo = transform(pneumo, let=log(exposure.time))
vgam(cbind(normal,mild,severe) ~ s(let), cumulative(par=TRUE), pneumo)

# Nonparametric logistic regression
data(hunua)
fit = vgam(agaaus ~ s(altitude, df=2), binomialff, hunua)
## Not run:
plot(fit, se=TRUE)
## End(Not run)

# Fit two species simultaneously
fit2 = vgam(cbind(agaaus, kniexc) ~ s(altitude, df=c(2,3)),
            binomialff(mv=TRUE), hunua)
coef(fit2, mat=TRUE)   # Not really interpretable
## Not run:
plot(fit2, se=TRUE, overlay=TRUE, lcol=1:2, scol=1:2)
attach(hunua)
o = order(altitude)
matplot(altitude[o], fitted(fit2)[o,], type="l", lwd=2, las=1,
    xlab="Altitude (m)", ylab="Probability of presence",
    main="Two plant species' response curves", ylim=c(0,.8))
rug(altitude)
detach(hunua)
## End(Not run)
```

---

vgam.control *Control function for vgam*

Description

Algorithmic constants and parameters for running vgam are set using this function.

Usage

```
vgam.control(all.knots = FALSE, backchat = if (is.R()) FALSE else TRUE,
             bf.epsilon = 1e-07, bf.maxit = 30,
             checkwz=TRUE,
             criterion = names(.min.criterion.VGAM),
             epsilon = 1e-07, maxit = 30, na.action = na.fail,
             nk = NULL, save.weight = FALSE, se.fit = TRUE,
             trace = FALSE, wzepsilon = .Machine$double.eps^0.75,
             xij = NULL, ...)
```

Arguments

In the following, we let d be the number of `s` terms in the formula.

`all.knots`	logical indicating if all distinct points of the smoothing variables are to be used as knots. By default, `all.knots=TRUE` for $n \leq 40$, and for $n > 40$, the number of knots is approximately $40 + (n - 40)^{0.25}$. This increases very slowly with n so that the number of knots is approximately between 50 and 60 for large n.
`backchat`	logical indicating if a backchat is to be used (not applicable in R).
`bf.epsilon`	tolerance used by the modified vector backfitting algorithm for testing convergence. Must be a positive number.
`bf.maxit`	maximum number of iterations allowed in the modified vector backfitting algorithm. Must be a positive integer.
`checkwz`	logical indicating whether the diagonal elements of the working weight matrices should be checked whether they are sufficiently positive, i.e., greater than `wzepsilon`. If not, any values less than `wzepsilon` are replaced with this value.
`criterion`	character variable describing what criterion is to be used to test for convergence. The possibilities are listed in `.min.criterion.VGAM`, but most family functions only implement a few of these.
`epsilon`	positive convergence tolerance epsilon. Roughly speaking, the Newton-Raphson/Fisher-scoring/local-scoring iterations are assumed to have converged when two successive `criterion` values are within `epsilon` of each other.
`maxit`	maximum number of Newton-Raphson/Fisher-scoring/local-scoring iterations allowed.
`na.action`	how to handle missing values. Unlike the SPLUS `gam` function, `vgam` cannot handle `NA`s when smoothing.
`nk`	vector of length d containing positive integers. Recycling is used if necessary. The ith value is the number of B-spline coefficients to be estimated for each component function of the ith `s()` term. `nk` differs from the number of knots by some constant. If specified, `nk` overrides the automatic knot selection procedure.
`save.weight`	logical indicating whether the `weights` slot of a `"vglm"` object will be saved on the object. If not, it will be reconstructed when needed, e.g., `summary`.
`se.fit`	logical indicating whether approximate pointwise standard errors are to be saved on the object. If `TRUE`, then these can be plotted with `plot(..., se=TRUE)`.
`trace`	logical indicating if output should be produced for each iteration.
`wzepsilon`	Small positive number used to test whether the diagonals of the working weight matrices are sufficiently positive.
`xij`	formula giving terms making up a covariate-dependent term.
`...`	other parameters that may be picked up from control functions that are specific to the **VGAM** family function.

Details

Most of the control parameters are used within `vgam.fit` and you will have to look at that to understand the full details. Many of the control parameters are used in a similar manner by `vglm.fit` (`vglm`) because the algorithm (IRLS) is very similar.

Setting `save.weight=FALSE` is useful for some models because the `weights` slot of the object is often the largest and so less memory is used to store the object. However, for some **VGAM** family function, it is necessary to set `save.weight=TRUE` because the `weights` slot cannot be reconstructed later.

Value

A list with components matching the input names. A little error checking is done, but not much. The list is assigned to the `control` slot of `vgam` objects.

Note

`vgam` does not implement half-stepsizing, therefore parametric models should be fitted with `vglm`. Also, `vgam` is slower than `vglm` too.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Wild, C. J. (1996) Vector generalized additive models. *Journal of the Royal Statistical Society, Series B, Methodological*, **58**, 481–493.

http://www.stat.auckland.ac.nz/~yee

See Also

`vgam`, `vsmooth.spline`, `vglm`.

Examples

```
data(pneumo)
pneumo = transform(pneumo, let=log(exposure.time))
vgam(cbind(normal, mild, severe) ~ s(let, df=3), multinomial,
     pneumo, trace=TRUE, eps=1e-4, maxit=10)
```

`vglm-class` *Class "vglm"*

Description

Vector generalized linear models.

Objects from the Class

Objects can be created by calls of the form `vglm(...)`.

Slots

In the following, M is the number of linear predictors.

extra: Object of class `"list"`; the `extra` argument on entry to `vglm`. This contains any extra information that might be needed by the family function.

family: Object of class `"vglmff"`. The family function.

iter: Object of class `"numeric"`. The number of IRLS iterations used.

predictors: Object of class `"matrix"` with M columns which holds the M linear predictors.

assign: Object of class `"list"`, from class `"vlm"`. This named list gives information matching the columns and the (LM) model matrix terms.

call: Object of class `"call"`, from class `"vlm"`. The matched call.

coefficients: Object of class `"numeric"`, from class `"vlm"`. A named vector of coefficients.

constraints: Object of class `"list"`, from class `"vlm"`. A named list of constraint matrices used in the fitting.

contrasts: Object of class `"list"`, from class `"vlm"`. The contrasts used (if any).

control: Object of class `"list"`, from class `"vlm"`. A list of parameters for controlling the fitting process. See `vglm.control` for details.

criterion: Object of class `"list"`, from class `"vlm"`. List of convergence criterion evaluated at the final IRLS iteration.

df.residual: Object of class `"numeric"`, from class `"vlm"`. The residual degrees of freedom.

df.total: Object of class `"numeric"`, from class `"vlm"`. The total degrees of freedom.

dispersion: Object of class `"numeric"`, from class `"vlm"`. The scaling parameter.

effects: Object of class `"numeric"`, from class `"vlm"`. The effects.

fitted.values: Object of class `"matrix"`, from class `"vlm"`. The fitted values. This may be missing or consist entirely of `NA`s, e.g., the Cauchy model.

misc: Object of class `"list"`, from class `"vlm"`. A named list to hold miscellaneous parameters.

model: Object of class `"data.frame"`, from class `"vlm"`. The model frame.

na.action: Object of class "list", from class "vlm". A list holding information about missing values.

offset: Object of class "matrix", from class "vlm". If non-zero, a M-column matrix of offsets.

post: Object of class "list", from class "vlm" where post-analysis results may be put.

preplot: Object of class "list", from class "vlm" used by plotvgam; the plotting parameters may be put here.

prior.weights: Object of class "numeric", from class "vlm" holding the initially supplied weights.

qr: Object of class "list", from class "vlm". QR decomposition at the final iteration.

R: Object of class "matrix", from class "vlm". The **R** matrix in the QR decomposition used in the fitting.

rank: Object of class "integer", from class "vlm". Numerical rank of the fitted model.

residuals: Object of class "matrix", from class "vlm". The *working* residuals at the final IRLS iteration.

rss: Object of class "numeric", from class "vlm". Residual sum of squares at the final IRLS iteration with the adjusted dependent vectors and weight matrices.

smart.prediction: Object of class "list", from class "vlm". A list of data-dependent parameters (if any) that are used by smart prediction.

terms: Object of class "list", from class "vlm". The terms object used.

weights: Object of class "matrix", from class "vlm". The weight matrices at the final IRLS iteration. This is in matrix-band form.

x: Object of class "matrix", from class "vlm". The model matrix (LM, not VGLM).

xlevels: Object of class "list", from class "vlm". The levels of the factors, if any, used in fitting.

y: Object of class "matrix", from class "vlm". The response, in matrix form.

Extends

Class "vlm", directly.

Methods

cdf signature(object = "vglm"): cumulative distribution function. Applicable to, e.g., quantile regression and extreme value data models.

deplot signature(object = "vglm"): Applicable to, e.g., quantile regression.

deviance signature(object = "vglm"): deviance of the model (where applicable).

plot signature(x = "vglm"): diagnostic plots.

predict signature(object = "vglm"): extract the linear predictors or predict the linear predictors at a new data frame.

print signature(x = "vglm"): short summary of the object.

qtplot signature(object = "vglm"): quantile plot (only applicable to some models).

resid `signature(object = "vglm")`: residuals. There are various types of these.

residuals `signature(object = "vglm")`: residuals. Shorthand for `resid`.

rlplot `signature(object = "vglm")`: return level plot. Useful for extreme value data models.

summary `signature(object = "vglm")`: a more detailed summary of the object.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

Yee, T. W. and Wild, C. J. (1996) Vector generalized additive models. *Journal of the Royal Statistical Society, Series B, Methodological*, **58**, 481–493.

http://www.stat.auckland.ac.nz/~yee

See Also

`vglm`, `vglmff-class`, `vgam-class`.

Examples

```
# Multinomial logit model
data(pneumo)
pneumo = transform(pneumo, let=log(exposure.time))
vglm(cbind(normal,mild,severe) ~ let, multinomial, pneumo)
```

---

vglm *Fitting Vector Generalized Linear Models*

Description

`vglm` is used to fit vector generalized linear models (VGLMs). This is a large class of models that includes generalized linear models (GLMs) as special cases.

Usage

```
vglm(formula, family, data = list(), weights = NULL, subset = NULL,
     na.action = na.fail, etastart = NULL, mustart = NULL,
     coefstart = NULL, control = vglm.control(...), offset = NULL,
     method = "vglm.fit", model = FALSE, x.arg = TRUE, y.arg = TRUE,
     contrasts = NULL, constraints = NULL, extra = list(),
     qr.arg = FALSE, smart = TRUE, ...)
```

Arguments

In the following, M is the number of linear predictors.

formula	a symbolic description of the model to be fit. The RHS of the formula is applied to each linear predictor. Different variables in each linear predictor can be chosen by specifying constraint matrices.
family	a function of class "vglmff" describing what statistical model is to be fitted. These are called "**VGAM** family functions".
data	an optional data frame containing the variables in the model. By default the variables are taken from environment(formula), typically the environment from which vglm is called.
weights	an optional vector or matrix of (prior) weights to be used in the fitting process. If weights is a matrix, then it must be in *matrix-band* form, whereby the first M columns of the matrix are the diagonals, followed by the upper-diagonal band, followed by the band above that, etc. In this case, there can be up to $M(M+1)$ columns, with the last column corresponding to the (1,M) elements of the weight matrices.
subset	an optional logical vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The "factory-fresh" default is na.omit.
etastart	starting values for the linear predictors. It is a M-column matrix. If $M = 1$ then it may be a vector.
mustart	starting values for the fitted values. It can be a vector or a matrix. Some family functions do not make use of this argument.
coefstart	starting values for the coefficient vector.
control	a list of parameters for controlling the fitting process. See vglm.control for details.
offset	a vector or M-column matrix of offset values. These are *a priori* known and are added to the linear predictors during fitting.
method	the method to be used in fitting the model. The default (and presently only) method vglm.fit uses iteratively reweighted least squares (IRLS).
model	a logical value indicating whether the *model frame* should be assigned in the model slot.
x.arg, y.arg	logical values indicating whether the model matrix and response vector/matrix used in the fitting process should be assigned in the x and y slots. Note the model matrix is the LM model matrix; to get the VGLM model matrix type model.matrix(vglmfit) where vglmfit is a vglm object.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
constraints	an optional list of constraint matrices. The components of the list must be named with the term it corresponds to (and it must match in character format exactly). Each constraint matrix must have M rows, and be of full-column rank. By

default, constraint matrices are the M by M identity matrix unless arguments in the family function itself override these values. If `constraints` is used it must contain *all* the terms; an incomplete list is not accepted.

`extra` an optional list with any extra information that might be needed by the **VGAM** family function.

`qr.arg` logical value indicating whether the slot `qr`, which returns the QR decomposition of the VLM model matrix, is returned on the object.

`smart` logical value indicating whether smart prediction (`smartpred`) will be used.

`...` further arguments passed into `vglm.control`.

Details

A vector generalized linear model (VGLM) is loosely defined as a statistical model that is a function of M linear predictors. The central formula is given by

$$\eta_j = \beta_j^T x$$

where x is a vector of explanatory variables (sometimes just a 1 for an intercept), and β_j is a vector of regression coefficients to be estimated. Here, $j = 1, \ldots, M$ where M is finite. Then one can write $\eta = (\eta_1, \ldots, \eta_M)^T$ as a vector of linear predictors.

Most users will find `vglm` similar in flavour to `glm`. The function `vglm.fit` actually does the work.

Value

An object of class `"vglm"`, which has the following slots. Some of these may not be assigned to save space, and will be recreated if necessary later.

`extra` the list `extra` at the end of fitting.

`family` the family function (of class `"vglmff"`).

`iter` the number of IRLS iterations used.

`predictors` a M-column matrix of linear predictors.

`assign` a named list which matches the columns and the (LM) model matrix terms.

`call` the matched call.

`coefficients` a named vector of coefficients.

`constraints` a named list of constraint matrices used in the fitting.

`contrasts` the contrasts used (if any).

`control` list of control parameter used in the fitting.

`criterion` list of convergence criterion evaluated at the final IRLS iteration.

`df.residual` the residual degrees of freedom.

`df.total` the total degrees of freedom.

`dispersion` the scaling parameter.

`effects` the effects.

`fitted.values`
: the fitted values, as a matrix. This may be missing or consist entirely of `NA`s, e.g., the Cauchy model.

`misc`
: a list to hold miscellaneous parameters.

`model`
: the model frame.

`na.action`
: a list holding information about missing values.

`offset`
: if non-zero, a M-column matrix of offsets.

`post`
: a list where post-analysis results may be put.

`preplot`
: used by `plotvgam`, the plotting parameters may be put here.

`prior.weights`
: initially supplied weights.

`qr`
: the QR decomposition used in the fitting.

`R`
: the **R** matrix in the QR decomposition used in the fitting.

`rank`
: numerical rank of the fitted model.

`residuals`
: the *working* residuals at the final IRLS iteration.

`rss`
: residual sum of squares at the final IRLS iteration with the adjusted dependent vectors and weight matrices.

`smart.prediction`
: a list of data-dependent parameters (if any) that are used by smart prediction.

`terms`
: the `terms` object used.

`weights`
: the weight matrices at the final IRLS iteration. This is in matrix-band form.

`x`
: the model matrix (linear model LM, not VGLM).

`xlevels`
: the levels of the factors, if any, used in fitting.

`y`
: the response, in matrix form.

This slot information is repeated at `vglm-class`.

Note

This function can fit a wide variety of statistical models. Some of these are harder to fit than others because of inherent numerical difficulties associated with some of them. Successful model fitting benefits from cumulative experience. Varying the values of arguments in the **VGAM** family function itself is a good first step if difficulties arise, especially if initial values can be inputted. A second, more general step, is to vary the values of arguments in `vglm.control`. A third step is to make use of arguments such as `etastart`, `coefstart` and `mustart`.

Some **VGAM** family functions end in `"ff"` to avoid interference with other functions, e.g., `binomialff`, `poissonff`, `gaussianff`, `gammaff`. This is because **VGAM** family functions are incompatible with `glm` (and also `gam` in the **gam** library and `gam` in the **mgcv** library).

The smart prediction (`smartpred`) library is packed with the **VGAM** library.

The theory behind the scaling parameter is currently being made more rigorous, but it it should give the same value as the scale parameter for GLMs.

In Example 5 below, the `xij` argument to illustrate covariates that are specific to a linear predictor. Here, `lop`/`rop` are the ocular pressures of the left/right eye (artificial data). Variables `leye` and `reye` might be the presence/absence of a particular disease on the LHS/RHS eye respectively. See `fill` for more details and examples.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

Yee, T. W. and Wild, C. J. (1996) Vector generalized additive models. *Journal of the Royal Statistical Society, Series B, Methodological*, **58**, 481–493.

The **VGAM** library can be downloaded starting from http://www.stat.auckland.ac.nz/~yee. Other **VGAM** resources and documentation can be found there.

See Also

vglm.control, vglm-class, vglmff-class, smartpred, vglm.fit, fill, rrvglm, vgam. Methods functions include coef.vlm, predict.vglm, summary.vglm, etc.

Examples

```
# Example 1. Dobson (1990) Page 93: Randomized Controlled Trial :
counts = c(18,17,15,20,10,20,25,13,12)
outcome = gl(3,1,9)
treatment = gl(3,3)
print(d.AD <- data.frame(treatment, outcome, counts))
vglm.D93 = vglm(counts ~ outcome + treatment, family=poissonff)
summary(vglm.D93)

# Example 2. Multinomial logit model
data(pneumo)
pneumo = transform(pneumo, let=log(exposure.time))
vglm(cbind(normal, mild, severe) ~ let, multinomial, pneumo)

# Example 3. Proportional odds model
fit = vglm(cbind(normal,mild,severe) ~ let, cumulative(par=TRUE), pneumo)
coef(fit, matrix=TRUE)
constraints(fit)
fit@x # LM model matrix
model.matrix(fit) # Larger VGLM model matrix

# Example 4. Bivariate logistic model
data(coalminers)
fit = vglm(cbind(nBnW, nBW, BnW, BW) ~ age, binom2.or, coalminers, trace=TRUE)
coef(fit, matrix=TRUE)
fit@y

# Example 5. The use of the xij argument
n = 1000
eyes = data.frame(lop = runif(n), rop = runif(n))
eyes = transform(eyes,
                 leye = ifelse(runif(n) < logit(-1+2*lop, inverse=TRUE), 1, 0),
                 reye = ifelse(runif(n) < logit(-1+2*rop, inverse=TRUE), 1, 0))
```

```
fit = vglm(cbind(leye,reye) ~ lop + rop + fill(lop),
           binom2.or(exchangeable=TRUE, zero=3),
           xij = op ~ lop + rop + fill(lop), data=eyes)
coef(fit)
coef(fit, matrix=TRUE)
coef(fit, matrix=TRUE, compress=FALSE)

# Here's one method to handle the xij argument with a term that
# produces more than one column in the model matrix.
POLY3 = function(x, ...) {
    # A cubic
    poly(c(x,...), 3)[1:length(x),]
}

fit = vglm(cbind(leye,reye) ~ POLY3(lop,rop) + POLY3(rop,lop) + fill(POLY3(lop,rop)),
           binom2.or(exchangeable=TRUE, zero=3),  data=eyes,
           xij = POLY3(op) ~ POLY3(lop,rop) + POLY3(rop,lop) +
                             fill(POLY3(lop,rop)))
coef(fit)
coef(fit, matrix=TRUE)
coef(fit, matrix=TRUE, compress=FALSE)
predict(fit)[1:4,]
```

`vglm.control` *Control function for vglm*

Description

Algorithmic constants and parameters for running `vglm` are set using this function.

Usage

```
vglm.control(backchat = if (is.R()) FALSE else TRUE,
             checkwz=TRUE, criterion = names(.min.criterion.VGAM),
             epsilon = 1e-07, half.stepsizing = TRUE,
             maxit = 30, stepsize = 1, save.weight = FALSE,
             trace = FALSE, wzepsilon = .Machine$double.eps^0.75,
             xij = NULL, ...)
```

Arguments

`backchat`	logical indicating if a backchat is to be used (not applicable in R).
`checkwz`	logical indicating whether the diagonal elements of the working weight matrices should be checked whether they are sufficiently positive, i.e., greater than `wzepsilon`. If not, any values less than `wzepsilon` are replaced with this value.
`criterion`	character variable describing what criterion is to be used to test for convergence. The possibilities are listed in `.min.criterion.VGAM`, but most family functions only implement a few of these.

`epsilon` positive convergence tolerance epsilon. Roughly speaking, the Newton-Raphson/Fisher-scoring iterations are assumed to have converged when two successive `criterion` values are within `epsilon` of each other.

`half.stepsizing` logical indicating if half-stepsizing is allowed. For example, in maximizing a log-likelihood, if the next iteration has a log-likelihood that is less than the current value of the log-likelihood, then a half step will be taken. If the log-likelihood is still less than at the current position, a quarter-step will be taken etc. Eventually a step will be taken so that an improvement is made to the convergence criterion. `half.stepsizing` is ignored if `criterion=="coefficients"`.

`maxit` maximum number of Newton-Raphson/Fisher-scoring iterations allowed.

`stepsize` usual step size to be taken between each Newton-Raphson/Fisher-scoring iteration. It should be a value between 0 and 1, where a value of unity corresponds to an ordinary step. A value of 0.5 means half-steps are taken. Setting a value near zero will cause convergence to be generally slow but may help increase the chances of successful convergence for some family functions.

`save.weight` logical indicating whether the `weights` slot of a `"vglm"` object will be saved on the object. If not, it will be reconstructed when needed, e.g., `summary`. Some family functions have `save.weight=TRUE` and others have `save.weight=FALSE` in their control functions.

`trace` logical indicating if output should be produced for each iteration.

`wzepsilon` Small positive number used to test whether the diagonals of the working weight matrices are sufficiently positive.

`xij` formula giving terms making up a covariate-dependent term (a variable that takes on different values for each linear/additive predictor. For example, the ocular pressure of each eye). There should be M unique terms; use `fill1`, `fill2`, `fill3`, etc. if necessary. Each formula should have a response which is taken as the name of that variable, and the terms are enumerated in sequential order. With more than one formula, use a list of formulas. See Example 2 below.

`...` other parameters that may be picked up from control functions that are specific to the **VGAM** family function.

Details

Most of the control parameters are used within `vglm.fit` and you will have to look at that to understand the full details.

Setting `save.weight=FALSE` is useful for some models because the `weights` slot of the object is the largest and so less memory is used to store the object. However, for some **VGAM** family function, it is necessary to set `save.weight=TRUE` because the `weights` slot cannot be reconstructed later.

Value

A list with components matching the input names. A little error checking is done, but not much. The list is assigned to the `control` slot of `vglm` objects.

Note

In Example 2 below there are two covariates that have linear/additive predictor specific values. These are handled using the `xij` argument.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

See Also

`vglm`, `fill`.

Examples

```
# Example 1.
data(pneumo)
pneumo = transform(pneumo, let=log(exposure.time))
vglm(cbind(normal,mild,severe) ~ let, multinomial, pneumo,
     crit="coef", step=0.5, trace=TRUE, eps=1e-8, maxit=40)

# Example 2. The use of the xij argument
set.seed(111)
n = 1000
ymat = rdiric(n, shape=c(4,7,3,1))
mydat = data.frame(x1=runif(n), x2=runif(n), x3=runif(n), x4=runif(n),
                   z1=runif(n), z2=runif(n), z3=runif(n), z4=runif(n))
mydat = round(mydat, dig=2)
fit = vglm(ymat ~ x1 + x2 + x3 + x4 + z1 + z2 + z3 + z4,
           fam = dirichlet, data=mydat, crit="c",
           xij = list(z ~ z1 + z2 + z3 + z4,
                      x ~ x1 + x2 + x3 + x4))
model.matrix(fit, type="lm")[1:7,]   # LM model matrix
model.matrix(fit, type="vlm")[1:7,]  # Big VLM model matrix
coef(fit)
coef(fit, matrix=TRUE)
coef(fit, matrix=TRUE, compress=FALSE)
max(abs(predict(fit)-predict(fit, new=mydat))) # Predicts correctly
summary(fit)
```

vglmff-class *Class "vglmff"*

Description

Family functions for the **VGAM** package

Objects from the Class

Objects can be created by calls of the form `new("vglmff", ...)`.

Slots

In the following, M is the number of linear/additive predictors.

blurb: Object of class `"character"` giving a small description of the model. Important arguments such as parameter link functions can be expressed here.

constraints: Object of class `"expression"` which sets up any constraint matrices defined by arguments in the family function. A `zero` argument is always fed into `cm.zero.vgam`, whereas other constraints are fed into `cm.vgam`.

deviance: Object of class `"function"` returning the deviance of the model. This slot is optional. If present, the function must have arguments `function(mu, y, w, residuals = FALSE, eta, extra=NULL)`. Deviance residuals are returned if `residuals = TRUE`.

fini: Object of class `"expression"` to insert code at a special position in `vglm.fit` or `vgam.fit`. This code is evaluated immediately after the fitting.

first: Object of class `"expression"` to insert code at a special position in `vglm` or `vgam`.

initialize: Object of class `"expression"` used to perform error checking (especially for the variable `y`) and obtain starting values for the model. In general, `etastart` or `mustart` are assigned values based on the variables `y`, `x` and `w`.

inverse: Object of class `"function"` which returns the fitted values, given the linear/additive predictors. The function must have arguments `function(eta, extra=NULL)`.

last: Object of class `"expression"` to insert code at a special position (at the very end) of `vglm.fit` or `vgam.fit`. This code is evaluated after the fitting. The list `misc` is often assigned components in this slot, which becomes the `misc` slot on the fitted object.

link: Object of class `"function"` which, given the fitted values, returns the linear/additive predictors. If present, the function must have arguments `function(mu, extra=NULL)`.

loglikelihood: Object of class `"function"` returning the log-likelihood of the model. This slot is optional. If present, the function must have arguments `function(mu, y, w, residuals = FALSE, eta, extra=NULL)`. The argument `residuals` can be ignored because log-likelihood residuals aren't defined.

middle: Object of class `"expression"` to insert code at a special position in `vglm.fit` or `vgam.fit`.

middle2: Object of class `"expression"` to insert code at a special position in `vglm.fit` or `vgam.fit`.

summary.dispersion: Object of class "logical" indicating whether the general VGLM formula (based on a residual sum of squares) can be used for computing the scaling/dispersion parameter. It is TRUE for most models except for nonlinear regression models.

vfamily: Object of class "character" giving class information about the family function. Although not developed at this stage, more flexible classes are planned in the future. For example, family functions sratio, cratio, cumulative, and acat all operate on categorical data, therefore will have a special class called "VGAMcat", say. Then if fit was a vglm object, then coef(fit) would print out the vglm coefficients plus "VGAMcat" information as well.

deriv: Object of class "expression" which returns a M-column matrix of first derivatives of the log-likelihood function with respect to the linear/additive predictors, i.e., the score vector. In Yee and Wild (1996) this is the $\boldsymbol{d}_i$ vector. Thus each row of the matrix returned by this slot is such a vector.

weight: Object of class "expression" which returns the second derivatives of the log-likelihood function with respect to the linear/additive predictors. This can be either the observed or expected information matrix, i.e., Newton-Raphson or Fisher-scoring respectively. In Yee and Wild (1996) this is the $\boldsymbol{W}_i$ matrix. Thus each row of the matrix returned by this slot is such a matrix. Like the weights slot of vglm/vgam, it is stored in *matrix-band* form, whereby the first M columns of the matrix are the diagonals, followed by the upper-diagonal band, followed by the band above that, etc. In this case, there can be up to $M(M+1)$ columns, with the last column corresponding to the (1,M) elements of the weight matrices.

Methods

print signature(x = "vglmff"): short summary of the family function.

Warning

VGAM family functions are not compatible with glm, nor gam (from either **gam** or **mgcv** packages).

Note

With link functions etc., one must use substitute to embed the options into the code. There are two different forms: eval(substitute(expression({...}), list(...))) for expressions, and eval(substitute(function(...) { ... }, list(...))) for functions.

A unified method of handling arguments is to use match.arg. This allows, for example, vglm(..., family = cratio(link=logit)) and vglm(..., family = cratio(link="logi")) to be equivalent (Nb. there is a logit function).

The extra argument in inverse, link, deviance, loglikelihood, etc. matches with the argument extra in vglm, vgam and rrvglm. This allows input to be fed into all slots of a **VGAM** family function.

The expression derivative is evaluated immediately prior to weight, so there is provision for re-use of variables etc. Programmers must be careful to choose variable names that do not interfere with vglm.fit, vgam.fit etc.

Programmers of **VGAM** family functions are encouraged to keep to previous conventions regarding the naming of arguments, e.g., `link` is the argument for parameter link functions, `zero` for allowing some of the linear/additive predictors to be an intercept term only, etc.

In general, Fisher-scoring is recommended over Newton-Raphson where tractable. Although usually slightly slower in convergence, the weight matrices from using the expected information are positive-definite over a larger parameter space.

Author(s)

Thomas W. Yee

References

Yee, T. W. and Wild, C. J. (1996) Vector generalized additive models. *Journal of the Royal Statistical Society, Series B, Methodological*, **58**, 481–493.

`http://www.stat.auckland.ac.nz/~yee` contains further information on how to write **VGAM** family functions. The file is amongst other **VGAM** PDF documentation.

See Also

`vglm`, `vgam`, `rrvglm`.

Examples

```
cratio()
cratio(link="cloglog")
cratio(link=cloglog, reverse=TRUE)
```

---

vonmises *von Mises Distribution Family Function*

Description

Estimates the location and scale parameters of the von Mises distribution by maximum likelihood estimation.

Usage

```
vonmises(llocation="elogit", lscale="loge",
      elocation=if(llocation=="elogit") list(min=0, max=2*pi) else list(),
      escale=list(),
      ilocation=NULL, iscale=NULL,
      method.init=1, zero=NULL)
```

Arguments

`llocation, lscale`
: Parameter link functions applied to the location a parameter and scale parameter k, respectively. See `Links` for more choices. For k, a log link is the default because the parameter is positive.

`elocation, escale`
: List. Extra argument for each of the link functions. See `earg` in `Links` for general information.

`ilocation`
: Initial value for the location a parameter. By default, an initial value is chosen internally using `method.init`. Assigning a value will override the argument `method.init`.

`iscale`
: Initial value for the scale k parameter. By default, an initial value is chosen internally using `method.init`. Assigning a value will override the argument `method.init`.

`method.init`
: An integer with value `1` or `2` which specifies the initialization method. If failure to converge occurs try the other value, or else specify a value for `ilocation` and `iscale`.

`zero`
: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The default is none of them. If used, choose one value from the set {1,2}.

Details

The (two-parameter) von Mises distribution has a density that can be written as

$$f(y; a, k) = \frac{\exp[k \cos(y - a)]}{2\pi I_0(k)}$$

where $0 \leq y < 2\pi$, $k > 0$ is the scale parameter, a is the location parameter, and $I_0(k)$ is the modified Bessel function of order 0 evaluated at k. The mean of Y (which is the fitted value) is a and the circular variance is $1 - I_1(k)/I_0(k)$ where $I_1(k)$ is the modified Bessel function of order 1. By default, $\eta_1 = \log(a/(2\pi - a))$ and $\eta_2 = \log(k)$ for this family function.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

Warning

Numerically, the von Mises can be difficult to fit because of a log-likelihood having multiple maxima. The user is therefore encouraged to try different starting values, i.e., make use of `ilocation` and `iscale`.

Note

The response and the fitted values are scaled so that $0 \leq y < 2\pi$. The linear/additive predictors are left alone. Fisher scoring is used.

Author(s)

T. W. Yee

References

Evans, M., Hastings, N. and Peacock, B. (2000) *Statistical Distributions*, New York: Wiley-Interscience, Third edition.

See Also

Bessel.

CircStats and **circular** currently have a lot more R functions for circular data than the **VGAM** package.

Examples

```
x = runif(n <- 1000)
y = rnorm(n, m=2+x, sd=exp(0.2))   # Not von Mises data!!
fit = vglm(y  ~ x, vonmises(zero=2), trace=TRUE)
coef(fit, matrix=TRUE)
Coef(fit)
range(y)       # original data
range(fit@y)   # processed data is in [0,2*pi)
```

---

vsmooth.spline *Vector cubic smoothing spline*

Description

Fits a vector cubic smoothing spline.

Usage

```
vsmooth.spline(x, y, w, df = rep(5, M), spar = NULL,
               all.knots = FALSE,
               iconstraint = diag(M), xconstraint = diag(M),
               constraints = list("(Intercepts)" = diag(M), x = diag(M)),
               tol.nl = 0.01, var.arg = FALSE, scale.w = TRUE, nk = NULL)
```

Arguments

x	A vector, matrix or a list. If a list, the x component is used. If a matrix, the first column is used. x may also be a complex vector, in which case the real part is used, and the imaginary part is used for the response. In this help file, n is the number of unique values of x.
y	A vector, matrix or a list. If a list, the y component is used. If a matrix, all but the first column is used. In this help file, M is the number of columns of y if there are no constraints on the functions.

w	The weight matrices or the number of observations. If the weight matrices, then this must be a n-row matrix with the elements in matrix-band form (see iam). If a vector, then these are the number of observations. By default, w is the M by M identity matrix, denoted by matrix(1, n, M).
df	Numerical vector containing the degrees of freedom for each component function (smooth). If necessary, the vector is recycled to have length equal to the number of component functions to be estimated (M if there are no constraints), which equals the number of columns of the x-constraint matrix. A value of 2 means a linear fit, and each element of df should lie between 2 and n. The larger the values of df the more wiggly the smooths.
spar	Numerical vector containing the non-negative smoothing parameters for each component function (smooth). If necessary, the vector is recycled to have length equal to the number of component functions to be estimated (M if there are no constraints), which equals the number of columns of the x-constraint matrix. A value of zero means the smooth goes through the data and hence is wiggly. A value of Inf may be assigned, meaning the smooth will be linear. By default, the NULL value of spar means df is used to determine the smoothing parameters.
all.knots	Logical. If TRUE then each distinct value of x will be a knot. By default, only a subset of the unique values of x are used; typically, the number of knots is O(n^0.25) for n large, but if n <= 40 then all the unique values of x are used.
iconstraint	A M-row constraint matrix for the intercepts. It must be of full column rank. By default, the constraint matrix for the intercepts is the M by M identity matrix, meaning no constraints.
xconstraint	A M-row constraint matrix for x. It must be of full column rank. By default, the constraint matrix for the intercepts is the M by M identity matrix, meaning no constraints.
constraints	An alternative to specifying iconstraint and xconstraint, this is a list with two components corresponding to the intercept and x respectively. They must both be a M-row constraint matrix with full column rank.
tol.nl	Tolerance for testing nonlinearity for the component functions. If df is within tol.nl of 2 then the function is treated as linear.
var.arg	Logical: return the pointwise variances of the fit? Currently, this corresponds only to the nonlinear part of the fit, and may be wrong.
scale.w	Logical. By default, the weights w are scaled so that the diagonal elements have mean 1.
nk	Number of knots. If used, this argument overrides all.knots, and must lie between 6 and n+2 inclusive.

Details

The algorithm implemented is detailed in Yee (2000). It involves decomposing the component functions into a linear and nonlinear part, and using B-splines. The cost of the computation is O(n M^3).

The argument spar contains *scaled* smoothing parameters.

Value

An object of class "vsmooth.spline" (see vsmooth.spline-class).

Note

This function is quite similar to smooth.spline. For M=1, the results will be generally different, mainly due to the different way the knots are selected.

The vector cubic smoothing spline which s() represents is computationally demanding for large M. The cost is approximately $O(M^3)$.

Yet to be done: return the *unscaled* smoothing parameters.

Author(s)

Thomas W. Yee

References

Yee, T. W. (2000) Vector Splines and Other Vector Smoothers. Pages 529–534. In: Bethlehem, J. G. and van der Heijde, P. G. M. *Proceedings in Computational Statistics COMPSTAT 2000*. Heidelberg: Physica-Verlag.

See Also

vsmooth.spline-class, plot.vsmooth.spline, predict.vsmooth.spline, iam, s, smooth.spline.

Examples

```
n = 20
x = 2 + 5*(n:1)/n
x[2:4] = x[5:7]      # Allow duplication
y1 = sin(x) + rnorm(n, sd=0.13)
y2 = cos(x) + rnorm(n, sd=0.13)
y3 = 1 + sin(x) + rnorm(n, sd=0.13)  # Run this for constraints
y = cbind(y1, y2, y3)
ww = cbind(rep(3,n), 4, (1:n)/n)

(fit = vsmooth.spline(x, y, w=ww, df=5))
## Not run:
plot(fit) # The 1st and 3rd functions do not differ by a constant
## End(Not run)

mat = matrix(c(1,0,1, 0,1,0), 3, 2)
(fit2 = vsmooth.spline(x, y, w=ww, df=5, iconstr=mat, xconstr=mat))
# The 1st and 3rd functions do differ by a constant:
mycols = c("red","blue","red")
## Not run:
plot(fit2, lcol=mycols, pcol=mycols, las=1)
## End(Not run)
```

```
p = predict(fit, x=fit@x, deriv=0)
max(abs(fit@y - p$y))  # Should be zero

par(mfrow=c(3,1))
ux = seq(1, 8, len=100)
for(d in 1:3) {
    p = predict(fit, x=ux, deriv=d)
## Not run:
    matplot(p$x, p$y, type="l", main=paste("deriv =", d), lwd=2)
## End(Not run)
}
```

`waitakere` *Waitakere Ranges Data*

Description

The `waitakere` data frame has 579 rows and 18 columns. Altitude is explanatory, and there are binary responses (presence/absence = 1/0 respectively) for 17 plant species.

Usage

```
data(waitakere)
```

Format

This data frame contains the following columns:

agaaus Agathis australis, or Kauri

beitaw Beilschmiedia tawa, or Tawa

corlae Corynocarpus laevigatus

cyadea Cyathea dealbata

cyamed Cyathea medullaris

daccup Dacrydium cupressinum

dacdac Dacrycarpus dacrydioides

eladen Elaecarpus dentatus

hedarb Hedycarya arborea

hohpop Species name unknown

kniexc Knightia excelsa, or Rewarewa

kuneri Kunzea ericoides

lepsco Leptospermum scoparium

metrob Metrosideros robusta

neslan Nestegis lanceolata

rhosap Rhopalostylis sapida

vitluc Vitex lucens, or Puriri

altitude meters above sea level

Details

These were collected from the Waitakere Ranges, a small forest in northern Auckland, New Zealand. At 579 sites in the forest, the presence/absence of 17 plant species was recorded, as well as the altitude. Each site was of area size $200m^2$.

Source

Dr Neil Mitchell, University of Auckland.

See Also

hunua.

Examples

```
data(waitakere)
fit = vgam(agaaus ~ s(altitude, df=2), binomialff, waitakere)
## Not run:
plot(fit, se=TRUE, lcol="red", scol="blue")
## End(Not run)
predict(fit, waitakere, type="response")[1:3]
```

---

wald *Wald Distribution Family Function*

Description

Estimates the parameter of the standard Wald distribution by maximum likelihood estimation.

Usage

```
wald(link.lambda="loge", earg=list(), init.lambda=NULL)
```

Arguments

link.lambda	Parameter link function for the λ parameter. See Links for more choices.
earg	List. Extra argument for the link. See earg in Links for general information.
init.lambda	Initial value for the λ parameter. The default means an initial value is chosen internally.

Details

The standard Wald distribution is a special case of the inverse Gaussian distribution with $\mu = 1$. It has a density that can be written as

$$f(y;\lambda) = \sqrt{\lambda/(2\pi y^3)}\ \exp\left(-\lambda(y-1)^2/(2y)\right)$$

where $y > 0$ and $\lambda > 0$. The mean of Y is 1 (returned as the fitted values) and its variance is $1/\lambda$. By default, $\eta = \log(\lambda)$.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

The **VGAM** family function `inv.gaussianff` estimates the location parameter μ too.

Author(s)

T. W. Yee

References

Johnson, N. L. and Kotz, S. and Balakrishnan, N. (1994) *Continuous Univariate Distributions*, 2nd edition, Volume 1, New York: Wiley.

See Also

`inv.gaussianff`.

Examples

```
set.seed(123)
shape = 1
y = rgamma(n=1000, shape=shape) # Not inverse Gaussian!!
fit = vglm(y ~ 1, wald(init=0.2), trace=TRUE)
coef(fit, matrix=TRUE)
Coef(fit)
summary(fit)
```

---

weibull *Weibull Distribution Family Function*

Description

Maximum likelihood estimation of the 2-parameter Weibull distribution. No observations should be censored.

Usage

```
weibull(lshape = "loge", lscale = "loge",
        eshape = list(), escale = list(),
        ishape = NULL, iscale = NULL,
        nrfs = 1, imethod=1, zero = 2)
```

Arguments

`lshape, lscale`
: Parameter link functions applied to the (positive) shape parameter (called a below) and (positive) scale parameter (called b below). See `Links` for more choices.

`eshape, escale`
: Extra argument for the respective links. See `earg` in `Links` for general information.

`ishape, iscale`
: Optional initial values for the shape and scale parameters.

`nrfs`
: Currently this argument is ignored. Numeric, of length one, with value in $[0, 1]$. Weighting factor between Newton-Raphson and Fisher scoring. The value 0 means pure Newton-Raphson, while 1 means pure Fisher scoring. The default value uses a mixture of the two algorithms, and retaining positive-definite working weights.

`imethod`
: Initialization method used if there are censored observations. Currently only the values 1 and 2 are allowed.

`zero`
: An integer specifying which linear/additive predictor is to be modelled as an intercept only. The value must be from the set {1,2}, which correspond to the shape and scale parameters respectively. Setting `zero=NULL` means none of them.

Details

The Weibull density for a response Y is

$$f(y; a, b) = a y^{a-1} \exp[-(y/b)^a]/(b^a)$$

for $a > 0$, $b > 0$, $y > 0$. The cumulative distribution function is

$$F(y; a, b) = 1 - \exp[-(y/b)^a].$$

The mean of Y is $b\,\Gamma(1 + 1/a)$ (returned as the fitted values), and the mode is at $b\,(1 - 1/a)^{1/a}$ when $a > 1$. The density is unbounded for $a < 1$. The kth moment about the origin is $E(Y^k) = b^k\,\Gamma(1 + k/a)$. The hazard function is $a t^{a-1}/b^a$.

This **VGAM** family function currently does not handle censored data. Fisher scoring is used to estimate the two parameters. Although the Fisher information matrices used here are valid in all regions of the parameter space, the regularity conditions for maximum likelihood estimation are satisfied only if $a > 2$ (according to Kleiber and Kotz (2003)). If this is violated then a warning message is issued. One can enforce $a > 2$ by choosing `lshape = "logoff"` and `eshape=list(offset=-2)`.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Warning

This function is under development to handle other censoring situations. The version of this function which will handle censored data will be called `cenweibull()`. It is currently being written and will use `Surv` as input. It should be released in later versions of **VGAM**.

If the shape parameter is less than two then misleading inference may result, e.g., in the `summary` and `vcov` of the object.

Note

Successful convergence depends on having reasonably good initial values. If the initial values chosen by this function are not good, make use the two initial value arguments.

The Weibull distribution is often an alternative to the lognormal distribution. The inverse Weibull distribution, which is that of $1/Y$ where Y has a Weibull(a, b) distribution, is known as the log-Gompertz distribution.

Author(s)

T. W. Yee

References

Kleiber, C. and Kotz, S. (2003) *Statistical Size Distributions in Economics and Actuarial Sciences*, Hoboken, NJ: Wiley-Interscience.

Johnson, N. L. and Kotz, S. and Balakrishnan, N. (1994) *Continuous Univariate Distributions*, 2nd edition, Volume 1, New York: Wiley.

Gupta, R. D. and Kundu, D. (2006) On the comparison of Fisher information of the Weibull and GE distributions, *Journal of Statistical Planning and Inference*, **136**, 3130–3144.

See Also

`dweibull`, `gev`, `lognormal`, `expexp`.

Examples

```
# Complete data
x = runif(n <- 1000)
y = rweibull(n, shape=exp(1+x), scale = exp(-0.5))
fit = vglm(y ~ x, weibull, trace=TRUE)
coef(fit, mat=TRUE)
vcov(fit)
summary(fit)
```

`weightsvglm` *Prior and Working Weights of a VGLM fit*

Description

Returns either the prior weights or working weights of a VGLM object.

Usage

```
weightsvglm(object, type = c("prior", "working"),
            matrix.arg = TRUE, ignore.slot = FALSE,
            deriv.arg = FALSE, ...)
```

Arguments

`object`	a model object from the **VGAM** R package that inherits from a *vector generalized linear model* (VGLM), e.g., a model of class `"vglm"`.
`type`	Character, which type of weight is to be returned? The default is the first one.
`matrix.arg`	Logical, whether the answer is returned as a matrix. If not, it will be a vector.
`ignore.slot`	Logical. If `TRUE` then `object@weights` is ignored even if it has been assigned, and the long calculation for `object@weights` is repeated. This may give a slightly different answer because of the final IRLS step at convergence may or may not assign the latest value of quantities such as the mean and weights.
`deriv.arg`	Logical. If `TRUE` then a list with components `deriv` and `weights` is returned. See below for more details.
`...`	Currently ignored.

Details

Prior weights are usually inputted with the `weights` argument in functions such as `vglm` and `vgam`. It may refer to frequencies of the individual data or be weight matrices specified beforehand.

Working weights are used by the IRLS algorithm. They correspond to the second derivatives of the log-likelihood function with respect to the linear predictors. The working weights correspond to positive-definite weight matrices and are returned in *matrix-band* form, e.g., the first M columns correspond to the diagonals, etc.

Value

If `type="working"` and `deriv=TRUE` then a list is returned with the two components described below. Otherwise the prior or working weights are returned depending on the value of `type`.

`deriv`	Typically the first derivative of the log-likelihood with respect to the linear predictors. For example, this is the variable `deriv.mu` in `vglm.fit()`, or equivalently, the matrix returned in the `"deriv"` slot of a **VGAM** family function.

`weights` The working weights.

Note

This function is intended to be similar to `weights.glm` (see `glm`).

Author(s)

Thomas W. Yee

References

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

Chambers, J. M. and T. J. Hastie (eds) (1992) *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

`glm`, `vglmff-class`, `vglm`.

Examples

```
data(pneumo)
pneumo = transform(pneumo, let=log(exposure.time))
(fit = vglm(cbind(normal, mild, severe) ~ let,
           cumulative(parallel=TRUE, reverse=TRUE), pneumo))
fit@y                      # These are sample proportions
weights(fit, type="prior", matrix=FALSE) # Number of observations

# Look at the working residuals
n = nrow(model.matrix(fit, type="lm"))
M = ncol(predict(fit))

temp = weights(fit, type="working", deriv=TRUE)
wz = m2adefault(temp$weights, M=M)  # In array format
wzinv = array(apply(wz, 3, solve), c(M,M,n))
wresid = matrix(NA, n, M)  # working residuals
for(i in 1:n)
    wresid[i,] = wzinv[,,i,drop=TRUE] %*% temp$deriv[i,]
max(abs(c(resid(fit, type="w")) - c(wresid))) # Should be 0

z = predict(fit) + wresid  # Adjusted dependent vector
z
```

wrapup.smart *Cleans Up After Smart Prediction*

Description

`wrapup.smart` deletes any variables used by smart prediction. Needed by both the modelling function and the prediction function.

Usage

```
wrapup.smart()
```

Details

The variables to be deleted are `.smart.prediction`, `.smart.prediction.counter`, and `.smart.prediction.mode`. The function `wrapup.smart` is useful in R because these variables are held in `smartpredenv`. In S-PLUS, `wrapup.smart` is not really necessary because the variables are placed in frame 1, which disappears when finished anyway.

References

See the technical help file at http://www.stat.auckland.ac.nz/~yee for details.

See Also

`setup.smart`.

Examples

```
## Not run:
# Place this inside modelling functions such as lm, glm, vglm.
wrapup.smart()     # Put at the end of lm
## End(Not run)
```

yeo.johnson *Yeo-Johnson Transformation*

Description

Computes the Yeo-Johnson transformation, which is a normalizing transformation.

Usage

```
yeo.johnson(y, lambda, derivative = 0,
            epsilon = sqrt(.Machine$double.eps), inverse = FALSE)
```

Arguments

y	Numeric, a vector or matrix.
lambda	Numeric. It is recycled to the same length as y if necessary.
derivative	Non-negative integer. The default is the ordinary function evaluation, otherwise the derivative with respect to lambda.
epsilon	Numeric and positive value. The tolerance given to values of lambda when comparing it to 0 or 2.
inverse	Logical. Return the inverse transformation?

Details

The Yeo-Johnson transformation can be thought of as an extension of the Box-Cox transformation. It handles both positive and negative values, whereas the Box-Cox transformation only handles positive values. Both can be used to transform the data so as to improve normality. They can be used to perform LMS quantile regression.

Value

The Yeo-Johnson transformation or its inverse, or its derivatives with respect to lambda, of y.

Note

If inverse=TRUE then the argument derivative=0 is required.

Author(s)

Thomas W. Yee

References

Yeo, I.-K. and Johnson, R. A. (2000) A new family of power transformations to improve normality or symmetry. *Biometrika*, **87**, 954–959.

Yee, T. W. (2004) Quantile regression via vector generalized additive models. *Statistics in Medicine*, **23**, 2295–2315.

See Also

lms.yjn, boxcox.

Examples

```
n = 200
y = seq(-4, 4, len=n)
ltry = c(0, 0.5, 1, 1.5, 2)  # Try these values of lambda
lltry = length(ltry)
psi = matrix(NA, n, lltry)
for(i in 1:lltry) {
    psi[,i] = yeo.johnson(y, lambda=ltry[i])
}
```

```
## Not run:
matplot(y, psi, type="l", ylim=c(-4, 4), lwd=2, lty=1:lltry,
        ylab="Yeo-Johnson transformation", col=1:lltry, las=1,
        main="Yeo-Johnson transformation with some values of lambda")
abline(v=0, h=0)
legend(x=c(1), y=c(-0.5), lty=1:lltry, legend=as.character(ltry),
       lwd=2, col=1:lltry)
## End(Not run)
```

`yip88` *Zero-Inflated Poisson Distribution (Yip (1988) algorithm)*

Description

Fits a zero-inflated Poisson distribution based on Yip (1988).

Usage

```
yip88(link.lambda = "loge", n.arg = NULL)
```

Arguments

`link.lambda`	Link function for the usual λ parameter. See `Links` for more choices.
`n.arg`	The total number of observations in the data set. Needed when the response variable has all the zeros deleted from it, so that the number of zeros can be determined.

Details

The method implemented here, Yip (1988), maximizes a *conditional* likelihood. Consequently, the methodology used here deletes the zeros from the data set, and is thus related to the positive Poisson distribution (where $P(Y = 0) = 0$).

The probability function of Y is 0 with probability ϕ, and Poisson(λ) with probability $1 - \phi$. Thus

$$P(Y = 0) = \phi + (1 - \phi)P(W = 0)$$

where W is Poisson(λ). The mean, $(1 - \phi)\lambda$, can be obtained by the extractor function `fitted` applied to the object.

This family function treats ϕ as a scalar. If you want to model both ϕ and λ as a function of covariates, try `zipoisson`.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

Warning

Under- or over-flow may occur if the data is ill-conditioned. Yip (1988) only considered ϕ being a scalar and not modelled as a function of covariates. To get around this limitation, try `zipoisson`.

Inference obtained from `summary.vglm` and `summary.vgam` may or may not be correct. In particular, the p-values, standard errors and degrees of freedom may need adjustment. Use simulation on artificial data to check that these are reasonable.

Note

The data may be inputted in two ways. The first is when the response is a vector of positive values, with the argument `n` in `yip88` specifying the total number of observations. The second is simply include all the data in the response. In this case, the zeros are trimmed off during the computation, and the `x` and `y` slots of the object, if assigned, will reflect this.

The estimate of ϕ is placed in the `misc` slot as `@misc$phi`. However, this estimate is computed only for intercept models, i.e., the formula is of the form `y ~ 1`.

Author(s)

Thomas W. Yee

References

Yip, P. (1988) Inference about the mean of a Poisson distribution in the presence of a nuisance parameter. *The Australian Journal of Statistics*, **30**, 299–306.

Angers, J-F. and Biswas, A. (2003) A Bayesian analysis of zero-inflated generalized Poisson model. *Computational Statistics & Data Analysis*, **42**, 37–46.

Documentation accompanying the **VGAM** package at `http://www.stat.auckland.ac.nz/~yee` contains further information and examples.

See Also

`zipoisson`, `Zipois`, `zapoisson`, `pospoisson`, `poissonff`, `dzipois`.

Examples

```
# Generate some artificial data
n = 1000
phi = 0.35
lambda = 2
y = rzipois(n, lambda, phi)
table(y)

# Two equivalent ways of fitting the same model
fit1 = vglm(y ~ 1, yip88(n=length(y)), subset=y>0, trace=TRUE, crit="c")
fit2 = vglm(y ~ 1, yip88, trace=TRUE, crit="c")
true.mean = (1-phi) * lambda
true.mean
mean(y)
fitted(fit1)[1:5,]
```

```
fit1@misc$phi      # The estimate of phi

# Compare the ZIP with the positive Poisson distribution
pp = vglm(y ~ 1, pospoisson, subset=y>0, tr=TRUE, crit="c")
coef(pp)
Coef(pp)
(coef(fit1) - coef(pp))   # Same
((fitted(fit1) - fitted(pp))[1:5,])   # Different

# Another example ------------------------------
y = 0:7    # Data from Angers and Biswas (2003)
w = c(182, 41, 12, 2, 2, 0, 0, 1)
y = y[w>0]
w = w[w>0]
yy = rep(y,w)

fit3 = vglm(yy ~ 1, yip88(n=length(yy)), subset=yy>0, trace=TRUE, crit="c")
fit3@misc$phi  # Estimate of phi (they get 0.5154 with standard error 0.0707)
coef(fit3, matrix=TRUE)
Coef(fit3)  # Estimate of lambda (they get 0.6997 with standard error 0.1520)
fitted(fit3)[1:5]
mean(yy) # compare this with fitted(fit3)
```

`zanegbinomial` *Zero-Altered Negative Binomial Distribution*

Description

Fits a zero-altered negative binomial distribution based on a conditional model involving a binomial distribution and a positive-negative binomial distribution.

Usage

```
zanegbinomial(lp0="logit", lmunb = "loge", lk = "loge",
              ep0=list(), emunb =list(), ek = list(),
              ik = 1, zero = -3, cutoff = 0.995, method.init=3)
```

Arguments

`lp0`	Link function for the parameter p_0, called `p0` here. See `Links` for more choices.
`lmunb`	Link function applied to the `munb` parameter, which is the mean μ_{nb} of an ordinary negative binomial distribution. See `Links` for more choices.
`lk`	Parameter link function applied to the reciprocal of the dispersion parameter, called `k`. That is, as `k` increases, the variance of the response decreases. See `Links` for more choices.
`ep0, emunb, ek`	List. Extra argument for the respective links. See `earg` in `Links` for general information.

ik	Initial values for k. They must be positive, and one value for each response/species.
zero	Integer valued vector, usually assigned -3 or 3 if used at all. Specifies which of the three linear predictors are modelled as an intercept only. By default, the k parameter (after lk is applied) for each response is modelled as a single unknown number that is estimated. It can be modelled as a function of the explanatory variables by setting zero=NULL. A negative value means that the value is recycled, so setting -3 means all k are intercept only.
cutoff	A numeric which is close to 1 but never exactly 1. Used to specify how many terms of the infinite series are actually used. The sum of the probabilites are added until they reach this value or more. It is like specifying p in an imaginary function qnegbin(p).
method.init	See negbinomial.

Details

The response Y is zero with probability p_0, or Y has a positive-negative binomial distribution with probability $1 - p_0$. Thus $0 < p_0 < 1$, which is modelled as a function of the covariates. The zero-altered negative binomial distribution differs from the zero-inflated negative binomial distribution in that the former has zeros coming from one source, whereas the latter has zeros coming from the negative binomial distribution too. The zero-inflated negative binomial distribution is currently not implemented in the **VGAM** package. Some people call the zero-altered negative binomial a *hurdle* model.

For one response/species, by default, the three linear/additive predictors are $(logit(p_0), \log(\mu_{nb}), \log(k))^T$. This vector is recycled for multiple species.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

The fitted.values slot of the fitted object, which should be extracted by the generic function fitted, returns the mean μ which is given by

$$\mu = (1 - p_0)\mu_{nb}/[1 - (k/(k + \mu_{nb}))^k].$$

Warning

Convergence for this **VGAM** family function seems to depend quite strongly on providing good initial values.

Inference obtained from summary.vglm and summary.vgam may or may not be correct. In particular, the p-values, standard errors and degrees of freedom may need adjustment. Use simulation on artificial data to check that these are reasonable.

Note

Note this family function allows p_0 to be modelled as functions of the covariates. It is a conditional model, not a mixture model.

This family function effectively combines posnegbinomial and binomialff into one family function.

This family function can handle a multivariate response, e.g., more than one species.

Author(s)

T. W. Yee

References

Welsh, A. H., Cunningham, R. B., Donnelly, C. F. and Lindenmayer, D. B. (1996) Modelling the abundances of rare species: statistical models for counts with extra zeros. *Ecological Modelling*, **88**, 297–308.

See Also

posnegbinomial, binomialff, rposnegbin, zipoisson.

Examples

```
## Not run:
x = runif(n <- 2000)
p0 = logit(-1 + 2*x, inverse=TRUE)
y1 = rposnegbin(n, munb=exp(0+2*x), k=exp(1))    # With covariates
y2 = rposnegbin(n, munb=exp(1+2*x), k=exp(1))    # With covariates
y1 = ifelse(runif(n) < p0, 0, y1)
y2 = ifelse(runif(n) < p0, 0, y2)
table(y1)
table(y2)

fit = vglm(cbind(y1,y2) ~ x, zanegbinomial, trace=TRUE)
coef(fit, matrix=TRUE)
fitted(fit)[1:9,]
predict(fit)[1:9,]
## End(Not run)
```

---

zapoisson *Zero-Altered Poisson Distribution*

Description

Fits a zero-altered Poisson distribution based on a conditional model involving a binomial distribution and a positive-Poisson distribution.

Usage

```
zapoisson(lp0 = "logit", llambda = "loge",
          ep0=list(), elambda=list(),
          zero=NULL)
```

Arguments

lp0	Link function for the parameter p_0, called p0 here. See Links for more choices.
llambda	Link function for the usual λ parameter. See Links for more choices.
ep0, elambda	Extra argument for the respective links. See earg in Links for general information.
zero	Integer valued vector, usually assigned -1 or 1 if used at all. Specifies which of the two linear/additive predictors are modelled as an intercept only. By default, both linear/additive predictors are modelled using the explanatory variables. If zero=1 then the p_0 parameter (after lp0 is applied) is modelled as a single unknown number that is estimated. It is modelled as a function of the explanatory variables by zero=NULL. A negative value means that the value is recycled, so setting -1 means all p_0 are intercept-only (for multivariate responses).

Details

The response Y is zero with probability p_0, or Y has a positive-Poisson(λ) distribution with probability $1 - p_0$. Thus $0 < p_0 < 1$, which is modelled as a function of the covariates. The zero-altered Poisson distribution differs from the zero-inflated Poisson distribution in that the former has zeros coming from one source, whereas the latter has zeros coming from the Poisson distribution too. Some people call the zero-altered Poisson a *hurdle* model.

For one response/species, by default, the two linear/additive predictors are $(logit(p_0), \log(\lambda))^T$.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

The fitted.values slot of the fitted object, which should be extracted by the generic function fitted, returns the mean μ which is given by

$$\mu = (1 - p_0)\lambda/[1 - \exp(-\lambda)].$$

Warning

Inference obtained from summary.vglm and summary.vgam may or may not be correct. In particular, the p-values, standard errors and degrees of freedom may need adjustment. Use simulation on artificial data to check that these are reasonable.

Note

There are subtle differences between this family function and zipoisson and yip88. In particular, zipoisson is a *mixture* model whereas zapoisson and yip88 are *conditional* models.

Note this family function allows p_0 to be modelled as functions of the covariates. It can be thought of an extension of yip88, which is also a conditional model but its ϕ parameter is a scalar only.

This family function effectively combines pospoisson and binomialff into one family function.

This family function can handle a multivariate response, e.g., more than one species.

Author(s)

T. W. Yee

References

Welsh, A. H., Cunningham, R. B., Donnelly, C. F. and Lindenmayer, D. B. (1996) Modelling the abundances of rare species: statistical models for counts with extra zeros. *Ecological Modelling*, **88**, 297–308.

Angers, J-F. and Biswas, A. (2003) A Bayesian analysis of zero-inflated generalized Poisson model. *Computational Statistics & Data Analysis*, **42**, 37–46.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

See Also

zipoisson, yip88, pospoisson, posnegbinomial, binomialff, rpospois.

Examples

```
x = runif(n <- 1000)
p0 = logit(-1 + 1*x, inverse=TRUE)
lambda = loge(-0.3 + 2*x, inverse=TRUE)
y = ifelse(runif(n) < p0, 0, rpospois(n, lambda))
table(y)
fit = vglm(y ~ x, zapoisson, trace=TRUE)
fit = vglm(y ~ x, zapoisson, trace=TRUE, crit="c")
fitted(fit)[1:5]
predict(fit)[1:5,]
predict(fit, untransform=TRUE)[1:5,]
coef(fit, matrix=TRUE)

# Another example ------------------------------
# Data from Angers and Biswas (2003)
y = 0:7;   w = c(182, 41, 12, 2, 2, 0, 0, 1)
y = y[w>0]
w = w[w>0]
yy = rep(y,w)
fit3 = vglm(yy ~ 1, zapoisson, trace=TRUE, crit="c")
coef(fit3, matrix=TRUE)
Coef(fit3)  # Estimate of lambda (they get 0.6997 with standard error 0.1520)
fitted(fit3)[1:5]
mean(yy) # compare this with fitted(fit3)
```

---

zero *The zero Argument in VGAM Family Functions*

Description

The `zero` argument allows users to conveniently model certain linear/additive predictors as intercepts only.

Details

Often a certain parameter needs to be modelled simply while other parameters in the model may be more complex, for example, the λ parameter in LMS-Box-Cox quantile regression should be modelled more simply compared to its μ parameter. Another example is the ξ parameter in a GEV distribution which is should be modelled simpler than its μ parameter. Using the `zero` argument allows this to be fitted conveniently without having to input all the constraint matrices explicitly.

The `zero` argument should be assigned an integer vector from the set {`1:M`} where `M` is the number of linear/additive predictors. Full details about constraint matrices can be found in the references.

Value

Nothing is returned. It is simply a convenient argument for constraining certain linear/additive predictors to be an intercept only.

Warning

The use of other arguments may conflict with the `zero` argument. For example, using `constraints` to input constraint matrices may conflict with the `zero` argument. Another example is the argument `parallel`. In general users should not assume any particular order of precedence when there is potential conflict of definition. Currently no checking for consistency is made.

The argument `zero` may be renamed in the future to something better.

Side Effects

The argument creates the appropriate constraint matrices internally.

Note

In all **VGAM** family functions `zero=NULL` means none of the linear/additive predictors are modelled as intercepts-only. Almost all **VGAM** family function have `zero=NULL` as the default, but there are some exceptions, e.g., `binom2.or`.

Typing something like `coef(fit, matrix=TRUE)` is a useful way to ensure that the `zero` argument has worked as expected.

Author(s)

T. W. Yee

References

Yee, T. W. and Wild, C. J. (1996) Vector generalized additive models. *Journal of the Royal Statistical Society, Series B, Methodological*, **58**, 481–493.

Yee, T. W. and Hastie, T. J. (2003) Reduced-rank vector generalized linear models. *Statistical Modelling*, **3**, 15–41.

http://www.stat.auckland.ac.nz/~yee

See Also

constraints.

Examples

```
args(multinomial)
args(binom2.or)
args(gpd)

#LMS quantile regression example
data(bminz)
fit = vglm(BMI ~ bs(age, df=4), fam=lms.bcg(zero=c(1,3)),
           data=bminz, trace=TRUE)
coef(fit, matrix=TRUE)
```

---

zeta *Riemann's Zeta Function*

Description

Computes Riemann's zeta function and its first two derivatives.

Usage

```
zeta(x, deriv = 0)
```

Arguments

x	A complex-valued vector/matrix whose real values must be ≥ 1. Otherwise, if x may be real. If deriv is 1 or 2 then x must be real and positive.
deriv	An integer equalling 0 or 1 or 2, which is the order of the derivative. The default means it is computed ordinarily.

Details

While the usual definition involves an infinite series, more efficient methods have been devised to compute the value. In particular, this function uses Euler-Maclaurin summation. Theoretically, the zeta function can be computed over the whole complex plane because of analytic continuation.

The formula used here for analytic continuation is

$$\zeta(s) = 2^s \pi^{s-1} \sin(\pi s/2) \Gamma(1-s) \zeta(1-s).$$

This is actually one of several formulas, but this one was discovered by Riemann himself and is called the *functional equation*.

Value

A vector/matrix of computed values.

Warning

This function has not been fully tested, especially the derivatives. In particular, analytic continuation does not work here for complex `x` with `Re(x)<1` because currently the `gamma` function does not handle complex arguments.

Note

Estimation of the parameter of the zeta distribution can be achieved with `zetaff`.

Author(s)

T. W. Yee, with the help of G. J. Tee.

References

Riemann, B. (1859) Ueber die Anzahl der Primzahlen unter einer gegebenen Grosse. *Monatsberichte der Berliner Akademie, November 1859*.

Edwards, H. M. (1974) *Riemann's Zeta Function*. Academic Press: New York.

Markman, B. (1965) The Riemann zeta function. *BIT*, **5**, 138–141.

Abramowitz, M. and Stegun, I. A. (1972) *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, New York: Dover Publications Inc.

See Also

`zetaff`, `lerch`, `gamma`.

Examples

```
zeta(2:10)

## Not run:
x = seq(1.2, 10, len=100)
plot(x, zeta(x), type="l", las=1, xlim=c(-12,10), ylim=c(-1,4), col="red")
x = seq(-12, 0.8, len=100)
lines(x, zeta(x), col="red")
abline(v=0, h=c(0,1), lty="dashed")

# Close up plot
x = seq(-14, -0.4, len=100)
plot(x, zeta(x), type="l", las=1, col="red")
abline(v=0, h=0, lty="dashed")

# Plot of the first derivatives
x = seq(0.04, 0.8, len=100)
plot(x, zeta(x, deriv=1), type="l", las=1, col="blue", xlim=c(0.04,3), ylim=c(-6,0))
x = seq(1.2, 3, len=100)
```

```
lines(x, zeta(x, deriv=1), col="blue")
abline(v=0, h=0, lty="dashed") ## End(Not run)

zeta(2) - pi^2 / 6      # Should be zero
zeta(4) - pi^4 / 90     # Should be zero
zeta(6) - pi^6 / 945    # Should be 0
zeta(8) - pi^8 / 9450   # Should be 0
# zeta(0, deriv=1) + 0.5 * log(2*pi) # Should be 0
```

`zetaff` *Zeta Distribution Family Function*

Description

Estimates the parameter of the zeta distribution.

Usage

```
zetaff(link = "loge", earg=list(), init.p = NULL)
```

Arguments

`link`	Parameter link function applied to the (positive) parameter p. See `Links` for more choices. Choosing `loglog` constrains $p > 1$, but may fail if the maximum likelihood estimate is less than one.
`earg`	List. Extra argument for the link. See `earg` in `Links` for general information.
`init.p`	Optional initial value for the parameter p. The default is to choose an initial value internally. If converge failure occurs use this argument to input a value.

Details

In this long tailed distribution the response must be a positive integer. The probability function for a response Y is

$$P(Y = y) = 1/[y^{p+1}\zeta(p+1)], \quad p > 0, \quad y = 1, 2, ...$$

where ζ is Riemann's zeta function. The parameter p is positive, therefore a log link is the default. The mean of Y is $\mu = \zeta(p)/\zeta(p+1)$ provided $p > 1$. The variance of Y is $\zeta(p-1)/\zeta(p+1) - \mu^2$ provided $p > 2$.

It appears that good initial values are needed for successful convergence. If convergence is not obtained, try several values ranging from values near 0 to values about 10 or more.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

Note

The `zeta` function may be used to compute values of the zeta function.

Author(s)

T. W. Yee

References

pp.465–471, Chapter 11 of Johnson N. L., Kotz S., and Kemp A. W. (1993) *Univariate Discrete Distributions*, 2nd ed. New York: Wiley.

Knight, K. (2000) *Mathematical Statistics*. Boca Raton: Chapman & Hall/CRC Press.

See Also

zeta, dzeta, hzeta, zipf.

Documentation accompanying the **VGAM** package at http://www.stat.auckland.ac.nz/~yee contains further information and examples.

Examples

```
y = 1:5     # Knight, p.304
w =  c(63, 14, 5, 1, 2)
fit = vglm(y ~ 1, zetaff, trace=TRUE, wei=w, crit="c")
(phat = Coef(fit)) # 1.682557
cbind(dzeta(y, phat) * sum(w), w)

weighted.mean(y, w)
fitted(fit, mat=FALSE)
predict(fit)

# MLE should satisfy the following:
mean(log(rep(y, w))) + zeta(1+phat, deriv=1)/zeta(1+phat) # Should be 0
```

Zibinom *Zero-Inflated Binomial Distribution*

Description

Density, distribution function, quantile function and random generation for the zero-inflated binomial distribution with parameter `phi`.

Usage

```
dzibinom(x, size, prob, log = FALSE, phi = 0)
pzibinom(q, size, prob, lower.tail = TRUE, log.p = FALSE, phi = 0)
qzibinom(p, size, prob, lower.tail = TRUE, log.p = FALSE, phi = 0)
rzibinom(n, size, prob, phi = 0)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`size`	number of trials. It is the N symbol in the formula given in `zibinomial`.
`prob`	probability of success on each trial.
`n`	number of observations. Must be a single positive integer.
`log, log.p, lower.tail`	Arguments that are passed on to `pbinom`.
`phi`	Probability of zero (ignoring the binomial distribution), called ϕ. The default value of $\phi = 0$ corresponds to the response having an ordinary binomial distribution.

Details

The probability function of Y is 0 with probability ϕ, and $Binomial(size, prob)$ with probability $1 - \phi$. Thus

$$P(Y = 0) = \phi + (1 - \phi)P(W = 0)$$

where W is distributed $Binomial(size, prob)$.

Value

`dzibinom` gives the density, `pzibinom` gives the distribution function, `qzibinom` gives the quantile function, and `rzibinom` generates random deviates.

Note

The argument `phi` is recycled to the required length, and must have values which lie in the interval [0,1].

Author(s)

Thomas W. Yee

See Also

`zibinomial`, `dbinom`.

Examples

```
prob = 0.2
size = 10
phi = 0.5
(i = dzibinom(0:size, size, prob, phi=phi))
cumsum(i) - pzibinom(0:size, size, prob, phi=phi)  # Should be 0s
table(rzibinom(100, size, prob, phi=phi))

table(qzibinom(runif(100), size, prob, phi=phi))
round(dzibinom(0:10, size, prob, phi=phi) * 100) # Should be similar
```

```
## Not run:
x = 0:size
plot(x, dzibinom(x, size, prob, phi=phi), type="h", ylab="Probability",
     main=paste("ZIB(", size, ", ", prob, ", phi=", phi, ") (blue) vs",
     " Binomial(", size, ", ", prob, ") (red and shifted slightly)", sep=""),
     lwd=2, col="blue", las=1)
lines(x+0.05, dbinom(x, size, prob), type="h", lwd=2, col="red")
## End(Not run)
```

`zibinomial` *Zero-Inflated Binomial Distribution Family Function*

Description

Fits a zero-inflated binomial distribution by maximum likelihood estimation.

Usage

```
zibinomial(lphi="logit", link.mu="logit",
           ephi=list(), emu=list(),
           iphi=NULL, zero=1, mv=FALSE)
```

Arguments

`lphi`	Link function for the parameter ϕ. See `Links` for more choices.
`link.mu`	Link function for the usual binomial probability μ parameter. See `Links` for more choices.
`ephi, emu`	List. Extra argument for the respective links. See `earg` in `Links` for general information.
`iphi`	Optional initial value for ϕ, whose value must lie between 0 and 1. The default is to compute an initial value internally.
`zero`	An integer specifying which linear/additive predictor is modelled as intercepts only. If given, the value must be either 1 or 2, and the default is the first. Setting `zero=NULL` enables both ϕ and μ to be modelled as a function of the explanatory variables.
`mv`	Logical. Currently it must be `FALSE` to mean the function does not handle multivariate responses. This is to remain compatible with the same argument in `binomialff`.

Details

This function uses Fisher scoring and is based on

$$P(Y = 0) = \phi + (1 - \phi)(1 - \mu)^N,$$

for $y = 0$, and

$$P(Y = y) = (1 - \phi)\binom{N}{Ny}\mu^{Ny}(1 - \mu)^{N(1-y)}.$$

for $y = 1/N, 2/N, \ldots, 1$. That is, the response is a sample proportion out of N trials, and the argument `size` in `rzibinom` is N here. The parameter ϕ satisfies $0 < \phi < 1$. The mean of Y is $E(Y) = (1 - \phi)\mu$ and these are returned as the fitted values. By default, the two linear/additive predictors are $(logit(\phi), logit(\mu))^T$.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Warning

Numerical problems can occur. Half-stepping is not uncommon. If failure to converge occurs, make use of the argument `iphi`.

Note

The response variable must have one of the formats described by `binomialff`, e.g., a factor or two column matrix or a vector of sample proportions with the `weights` argument specifying the values of N.

To work well, one needs $N > 1$ and $\mu > 0$, i.e., the larger N and μ are, the better.

For intercept-models and constant N over the n observations, the `misc` slot has a component called `p0` which is the estimate of $P(Y = 0)$. This family function currently cannot handle a multivariate response (only `mv=FALSE` can be handled).

Author(s)

T. W. Yee

See Also

`rzibinom`, `binomialff`, `posbinomial`, `rbinom`.

Examples

```
size = 10  # number of trials; N in the notation above
n = 200
phi = logit(0,inv=TRUE)  # 0.50
mubin = logit(-1,inv=TRUE) # Mean of an ordinary binomial distribution
sv = rep(size, len=n)
y = rzibinom(n=n, size=sv, prob=mubin, phi=phi) / sv # A proportion
table(y)
fit = vglm(y ~ 1, zibinomial, weight=sv, trace=TRUE)
coef(fit, matrix=TRUE)
Coef(fit) # Useful for intercept-only models
fit@misc$p0  # Estimate of P(Y=0)
fitted(fit)[1:4,]
```

```
mean(y) # Compare this with fitted(fit)
summary(fit)
```

`zipf` *Zipf Distribution Family Function*

Description

Estimates the parameter of the Zipf distribution.

Usage

```
zipf(N=NULL, link="loge", earg=list(), init.s=NULL)
```

Arguments

`N`	Number of elements, an integer satisfying `1 < N < Inf`. The default is to use the maximum value of the response. If given, `N` must be no less that the largest response value. If `N=Inf` and $s > 1$ then this is the zeta distribution (use `zetaff` instead).
`link`	Parameter link function applied to the (positive) parameter s. See `Links` for more choices.
`earg`	List. Extra argument for the link. See `earg` in `Links` for general information.
`init.s`	Optional initial value for the parameter s. The default is to choose an initial value internally. If converge failure occurs use this argument to input a value.

Details

The probability function for a response Y is

$$P(Y = y) = y^{-s} / \sum_{i=1}^{N} i^{-s}, \quad s > 0, \quad y = 1, 2, \ldots, N,$$

where s is the exponent characterizing the distribution. The mean of Y, which are returned as the fitted values, is $\mu = H_{N,s-1}/H_{N,s}$ where $H_{n,m} = \sum_{i=1}^{n} i^{-m}$ is the nth generalized harmonic number.

Zipf's law is an experimental law which is often applied to the study of the frequency of words in a corpus of natural language utterances. It states that the frequency of any word is inversely proportional to its rank in the frequency table. For example, "the" and "of" are first two most common words, and Zipf's law states that "the" is twice as common as "of". Many other natural phenomena conform to Zipf's law.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Note

Upon convergence, the `N` is stored as `@misc$N`.

Author(s)

T. W. Yee

References

pp.465–471, Chapter 11 of Johnson N. L., Kotz S., and Kemp A. W. (1993) *Univariate Discrete Distributions*, 2nd ed. New York: Wiley.

See Also

`dzipf`, `zetaff`.

Examples

```
y = 1:5; w = c(63, 14, 5, 1, 2)
fit = vglm(y ~ 1, zipf, trace=TRUE, weight=w)
fit = vglm(y ~ 1, zipf(link=identity, init=3.4), tra=TRUE, weight=w, cri="c")
fit@misc$N
(shat = Coef(fit))
weighted.mean(y, w)
fitted(fit, mat=FALSE)
```

`Zipf` *The Zipf Distribution*

Description

Density, and cumulative distribution function for the Zipf distribution.

Usage

```
dzipf(x, N, s)
pzipf(q, N, s)
```

Arguments

`x, q`	vector of quantiles.
`N, s`	the number of elements, and the exponent characterizing the distribution. See `zipf` for more details.

Details

This is a finite version of the zeta distribution. See `zipf` for more details.

Value

`dzipf` gives the density, and `pzipf` gives the cumulative distribution function.

Author(s)

T. W. Yee

See Also

`zipf`.

Examples

```
## Not run:
N = 10; s=0.5; y = 1:N
proby = dzipf(y, N=N, s=s)
plot(y, proby, type="h", col="blue", ylab="P[Y=y]", ylim=c(0,0.2),
     main=paste("Zipf(N=",N,", s=",s,")", sep=""))
sum(proby)  # Should be 1
max(abs(cumsum(proby) - pzipf(y, N=N, s=s)))  # Should be 0
## End(Not run)
```

Zipois *Zero-Inflated Poisson Distribution*

Description

Density, distribution function, quantile function and random generation for the zero-inflated Poisson distribution with parameter `phi`.

Usage

```
dzipois(x, lambda, phi = 0)
pzipois(q, lambda, phi = 0)
qzipois(p, lambda, phi = 0)
rzipois(n, lambda, phi = 0)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. Must be a single positive integer.
`lambda`	Vector of positive means.
`phi`	Probability of zero (ignoring the Poisson distribution), called ϕ. The default value of $\phi = 0$ corresponds to the response having an ordinary Poisson distribution.

Details

The probability function of Y is 0 with probability ϕ, and $Poisson(\lambda)$ with probability $1-\phi$. Thus

$$P(Y=0) = \phi + (1-\phi)P(W=0)$$

where W is distributed $Poisson(\lambda)$.

Value

`dzipois` gives the density, `pzipois` gives the distribution function, `qzipois` gives the quantile function, and `rzipois` generates random deviates.

Note

The argument `phi` is recycled to the required length, and must have values which lie in the interval [0,1].

Author(s)

Thomas W. Yee

See Also

`dpois`, `zipoisson`.

Examples

```
lambda = 4
phi = 0.8
(i = dzipois(0:7, lambda, phi))
cumsum(i) - pzipois(0:7, lambda, phi)  # Should be 0s
table(rzipois(100, lambda, phi))

table(qzipois(runif(100), lambda, phi))
round(dzipois(0:10, lambda, phi) * 100) # Should be similar

## Not run:
x = 0:10
plot(x, dzipois(x, lambda, phi), type="h", ylab="Probability",
     main=paste("ZIP(", lambda, ", phi=", phi, ") (blue) vs",
     " Poisson(", lambda, ") (red and shifted slightly)", sep=""),
     lwd=2, col="blue", las=1)
lines(x+0.05, dpois(x, lambda), type="h", lwd=2, col="red")
## End(Not run)
```

`zipoisson` *Zero-Inflated Poisson Distribution Family Function*

Description

Fits a zero-inflated Poisson distribution using full maximum likelihood estimation.

Usage

```
zipoisson(lphi="logit", llambda = "loge",
          ephi=list(), elambda =list(),
          iphi = NULL, method.init=1, shrinkage.init=0.8, zero = NULL)
```

Arguments

`lphi` Link function for the parameter ϕ. See `Links` for more choices.

`llambda` Link function for the usual λ parameter. See `Links` for more choices.

`ephi, elambda` List. Extra argument for the respective links. See `earg` in `Links` for general information.

`iphi` Optional initial value for ϕ, whose value must lie between 0 and 1. The default is to compute an initial value internally.

`method.init` An integer with value `1` or `2` which specifies the initialization method for λ. If failure to converge occurs try another value and/or else specify a value for `shrinkage.init` and/or else specify a value for `iphi`.

`shrinkage.init` How much shrinkage is used when initializing λ. The value must be between 0 and 1 inclusive, and a value of 0 means the individual response values are used, and a value of 1 means the median or mean is used. This argument is used in conjunction with `method.init`.

`zero` An integer specifying which linear/additive predictor is modelled as intercepts only. If given, the value must be either 1 or 2, and the default is none of them. Setting `zero=1` makes ϕ a single parameter.

Details

This function uses Fisher scoring and is based on

$$P(Y = 0) = \phi + (1 - \phi)\exp(-\lambda),$$

and for $y = 1, 2, \ldots$,

$$P(Y = y) = (1 - \phi)\exp(-\lambda)\lambda^y/y!.$$

The parameter ϕ satisfies $0 < \phi < 1$. The mean of Y is $(1 - \phi)\lambda$ and these are returned as the fitted values. By default, the two linear/additive predictors are $(logit(\phi), \log(\lambda))^T$.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

Numerical problems can occur. Half-stepping is not uncommon. If failure to converge occurs, try using combinations of method.init, shrinkage.init, iphi, and/or zero=1 if there are explanatory variables.

Note

For intercept-models, the misc slot has a component called p0 which is the estimate of $P(Y = 0)$. Note that $P(Y = 0)$ is not the parameter ϕ. This family function currently cannot handle a multivariate response.

This family function is now recommended above yip88.

The zero-*deflated* Poisson distribution cannot be handled with this family function. It can be handled with the zero-altered Poisson distribution; see zapoisson.

Author(s)

T. W. Yee

References

Thas, O. and Rayner, J. C. W. (2005) Smooth tests for the zero-inflated Poisson distribution. *Biometrics*, **61**, 808–815.

Data: Angers, J-F. and Biswas, A. (2003) A Bayesian analysis of zero-inflated generalized Poisson model. *Computational Statistics & Data Analysis*, **42**, 37–46.

See Also

zapoisson, Zipois, yip88, rpois.

Examples

```
x = runif(n <- 2000)
phi = logit(-0.5 + 1*x, inverse=TRUE)
lambda = loge(0.5 + 2*x, inverse=TRUE)
y = rzipois(n, lambda, phi)
table(y)
fit = vglm(y ~ x, zipoisson, trace=TRUE)
coef(fit, matrix=TRUE)  # These should agree with the above values

# Another example: data from McKendrick (1926).
y = 0:4  # Number of cholera cases per household in an Indian village
w = c(168, 32, 16, 6, 1)  # Frequencies; there are 223=sum(w) households
fit = vglm(y ~ 1, zipoisson, wei=w, trace=TRUE)
coef(fit, matrix=TRUE)
```

```
cbind(actual=w, fitted=
      dzipois(y, lambda=Coef(fit)[2], phi=Coef(fit)[1]) * sum(w))

# Another example: data from Angers and Biswas (2003)
y = 0:7
w = c(182, 41, 12, 2, 2, 0, 0, 1)
y = y[w>0]
w = w[w>0]
fit = vglm(y ~ 1, zipoisson(lphi=probit, iphi=0.3), wei=w, tra=TRUE)
fit@misc$prob0  # Estimate of P(Y=0)
coef(fit, matrix=TRUE)
Coef(fit)  # Estimate of phi and lambda
fitted(fit)
weighted.mean(y,w) # Compare this with fitted(fit)
summary(fit)
```

Index